COMPUTATIONAL METHODS FOR STATIC ALLOCATION AND REAL-TIME REDEPLOYMENT OF AMBULANCES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Mateo Restrepo

August 2008

COMPUTATIONAL METHODS FOR STATIC ALLOCATION AND REAL-TIME REDEPLOYMENT OF AMBULANCES

Mateo Restrepo, Ph.D.

Cornell University 2008

We propose new approaches to tackle the problems of static and dynamic ambulance fleet allocation.

Static ambulance fleet allocation refers to deciding on home bases for ambulances, to which they return after serving calls. The number and location of bases are given. The goal is to keep response times to calls as small as possible. The first part introduces two models for this problem. Both of them are based on the Erlang loss formula. The first model is stylized and serves to illustrate that allocating ambulances to bases in proportion to base offered load is often non-optimal. The second model is similar in spirit to queueing theoretical models developed in the past but uses the Erlang loss function as a key ingredient. A careful computational comparison shows that the predictions obtained from our model are often more accurate than those produced by previous models, especially in low utilization regimes. This model can be used as a prescreening tool to find promising candidate allocations to be further evaluated through detailed simulation.

Dynamic redeployment concerns the real-time relocation of idle ambulances so as to ensure better preparedness. The second part of this dissertation formulates this problem as a dynamic program in a high-dimensional and uncountable state space, and then resorts to approximate dynamic programing (ADP) techniques to obtain approximate solutions. To this end, a specially tailored

approximation architecture for the problem is developed. The architecture depends on a small number of free parameters which are tuned using simulated cost trajectories of the system and linear regression. Computational experiments show that the relocation policies obtained from this approach offer significant performance improvements relative to benchmark static-relocation policies.

In the third part we use the linear programming approach to ADP on the dynamic-redeployment problem, using the previously developed approximation architecture. We conclude that, although the policies obtained are comparable in quality to those obtained using regression, there are serious issues related to numerical stability. Furthermore, the amount of computation required makes this approach less practical than the regression-based one.

BIOGRAPHICAL SKETCH

Mateo was born in the glorious "Nueva Villa de Nuestra Señora de la Candelaria de Medellín" on September 10th of 1979, exactly 123 years after Elias Howe got a patent for the sewing machine. A few months later, and contradicting the pious origin of his name, Mateo would be baptized in a makeshift ritual by one of his father's comrades by sustained immersion in the numbing cold waters of a turbulent creek in the vicinity of his hometown. This primordial experience might be the key to explaining some of our hero's main physical and psychological qualities, such as the pale purple hue of his skin and the euphemistically termed crackpottedness of his character.

Since early age, Mateo's innate beatitude and virtue naturally kept his diet away from green vegetables and herbs in general. To parody his carnivorous inclinations, at the age of three, Mateo dressed himself in a custom bunny costume bearing embroidered cloth carrots. His keen sense of fashion allowed little Teo to differentiate himself from his dissolute fellow toddlers. While attending *Kindergarten* our paladin was famous for always donning very short shorts and a pair of Venetian red rubber boots. At the age of 5, his precocious smoothness allowed him to espouse a petite curly redheaded girl. Mateo's heathen and juvenile condition obliged him to perform this by means of a pagan ceremony around a tree using aluminium foil pieces as wedding rings. This relationship would be the first in a yet-to-end series of lascivious encounters with carmine-haired debauchees. During high school, Mateo was famous for his albescence and was therefore nicknamed as 'Agua Mala' (lit. 'Bad Water', Col. colloq. 'Jelly Fish'.) Among his hobbies during this period we can recount: funambulism, papiroflexia, cartomancy, omphaloskepsis and shark shaking.

After finishing a lurid period of high education at the "Universidad Nacional

de Colombia", Mateo went on towards a more sublime level of education in the U.S.A, a.k.a. "The Land of Freedom". This transition implied his transubstantiation from a torpid physicist to an ebullient applied mathematician concerned mostly with the ontological status of the concept of true randomness, the viability of astrology as a substitute for economic forecasting, and whatnot. These he pursued while studying combinatorial optimization as an entremets.

During grad school, Mateo was always displeased by galling allusions to ancient traditions of his native country such as presenting friends a "Colombian neck-tie", shaking one's derrière while dancing and dressing fried chicken with honey.

During his 60 moons at the Jewel of the Ivy League, Mateo was constantly besieged by sinful and bibulous feminines – many of them scarlet haired – who wanted to luxuriate in his exuberant company and mesmeric charm, and presumably bear his offspring, too. Our hero, in his chastity and integrity, never succubi to the temptation of these succumb or to the scents originated in the spirituous beverages that customarily accompanied them. Neither did he attend the clamors of the cohort of debauched wrongdoers and yeggs who called themselves his friends. Instead of getting involved in the Saturnalia that the Olympian city of Ithaca became on Saturdays, our Ulysses dedicated all of his efforts to the improvement of his soul and virtue. This he accomplished by drudging day and night, with the aide of his two hoary mentors, on the wearisome completion of his *magnum opus* which the base reader now holds in his vile hands.

Having finished his Ph.D., our generous hero shall set forth towards yet more dangerous seas. As part of his ever continuing struggle for the happiness and the well-being of the whole human race, in September 2008 he will begin

work in the cesspool of Wall Street, thereby risking his spiritual taintlessness in order to save the world from an imminent financial collapse.

To the human race.

ACKNOWLEDGEMENTS

> What the world needs is more geniuses with humility, there are so few of us left.
>
> Oscar Levant

First and foremost, I would like to thank both co-chairs of my committee, Professors Shane Henderson and Huseyin Topaloglu for their time, patience, wise advice on the research, and dedication to proof-reading and massively improving the paper manuscripts. Without them, I could not have brought this dissertation to fruitful completion. I extend my thanks to Professor Rick Durrett, who showed me the purer side of applied mathematics and generously allowed me to work with him and write proofs for one year and later light-heartedly let me go. Thanks to Professor Paat Rusmevichientong, for his enthusiasm and encouragement regarding this work and for agreeing to be last player in this dream team of a committee.

Thanks to Alex Currell, Armann Ingolfsson and Andrew Mason for the data used for the numerical experiments in Sections 2.5, 3.5, 3.6.

My gratitude goes also to Dolores Pendell for her invaluable administrative and human support during my five years at the Center for Applied Math (C.A.M.). Other people at C.A.M that deserve acknowledgment for their academic and personal support all this time are (in no particular order): Johnny Guzman, Emilia Huerta-Sánchez, Fergal Casey, Yannet Interian Fernández, Marcel Blais, Erik Sherwood, Chris Scheper, June Andrews, Diarmuid Cahalane.

I am greatly indebted to Vanja Dukić for her friendship and support throughout all these years and for proof-reading some of the early versions of the first

chapter.

I would like to thank the Colombian Student's association at C.U., for making many meals in Ithaca yummier and alleviating the cold of many Fridays. Thanks to Natalia Moreno for forcing me to have proper lunch so many times, and also for her kind company and encouragement during the latest stages of preparation of this manuscript.

Finally, I should say Thanks to the faculty and staff Computer Science department at C.U., for hiring me to work as a T.A. almost every semester during the past 4 years. The CS Department was the author's main source of funding and this work would not have been at all possible without it.

This manuscript is based upon work supported in part by the National Science Foundation Grants No. DMI 0400287 and DMI 0422133. All opinions, findings, and conclusions or recommendations expressed in this manuscript are those of the authors and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

Biographical Sketch . . . iii
Dedication . . . vi
Acknowledgements . . . vii
Table of Contents . . . ix
List of Tables . . . xi
List of Figures . . . xii
List of Abbreviations . . . xiii

1 Introduction **1**
1.1 Static ambulance allocation . . . 2
1.2 Dynamic programming based real time redeployment of ambulances . . . 6

2 Erlang loss models for the static deployment of ambulances **10**
2.1 Literature Review . . . 11
2.2 The Island Model . . . 13
2.3 Insights from the Island Model . . . 16
2.4 The Overflow Model . . . 20
2.5 Computational Results for the Overflow Model, the A-Hypercube Model and Simulation . . . 24
2.6 Conclusions and Future Research . . . 33
2.7 Acknowledgments . . . 34

3 Approximate dynamic programming for ambulance redeployment **36**
3.1 Literature review . . . 37
3.2 Ambulance Redeployment as a Markov Decision Process . . . 40
3.2.1 State Space . . . 40
3.2.2 Controls . . . 42
3.2.3 Fundamental Dynamics . . . 43
3.2.4 Transition Costs . . . 44
3.2.5 Objective Function and Optimality Equation . . . 45
3.3 Approximate Dynamic Programming . . . 47
3.4 Basis Functions . . . 50
3.4.1 Baseline . . . 50
3.4.2 Unreachable Calls . . . 51
3.4.3 Uncovered Call Rate . . . 51
3.4.4 Missed Call Rate . . . 52
3.4.5 Future Uncovered Call Rate . . . 54
3.4.6 Future Missed Call Rate . . . 55
3.5 Computational Results on Edmonton . . . 56
3.5.1 Experimental Setup . . . 56
3.5.2 Baseline Performance . . . 58

3.5.3 Comparison with Random Search 60
3.5.4 Making Additional Redeployments 62
3.6 Computational Results on a Second Metropolitan Area 64
3.6.1 Experimental Setup 65
3.6.2 Baseline Performance 66
3.6.3 Effect of Turn-Out Time 67
3.6.4 Varying Call Arrival Rates and Fleet Sizes 69
3.7 Conclusions 71

4 Linear programming based approximate dynamic programing for ambulance redeployment 73
4.1 Preliminaries 74
4.2 LP-based DP – exact and approximate formulations 75
4.2.1 LP-based Exact Dynamic Programming 76
4.2.2 LP-based Approximate Dynamic Programming 77
4.3 Implementation and Testing 81
4.3.1 Improving stability 85
4.4 Evaluation of RLP policies for making additional redeployments 91
4.5 Comparison with the Approximate Policy Iteration Approach . . 92
4.6 Conclusions and Directions for Further Research 94

A Detailed Derivation of Approximation Procedure 96

B Initializing the Iterative Procedure for Solving the Overflow Model Fixed-Point Equations 100

C Detailed Definition of Function $R(\cdot)$ 101

LIST OF TABLES

2.1 Comparison of accuracy measures of the overflow and A-Hypercube models for a range of average utilizations. An $*$ indicates an instance where a measure for a model is significantly better than the same measure for the other model. 30
2.2 Error statistics for individual vehicle utilizations according to the two models. 31

3.1 Effect of turn-out time on the performance gap between ADP and the benchmark policy. 68

LIST OF FIGURES

2.1 Optimal solutions to Problem (2.1)-(2.4) for different values of (λ_1, λ_2). 17
2.2 Value of n_1 in the optimal solutions $(n_1, N - n_1)$ to Problem (2.1)-(2.4) for different values of $\lambda_1/(\lambda_1 + \lambda_2)$. 18
2.3 Rate of lost calls for proportional allocation and the optimal allocation, for $\rho = 0.15, 0.3$ and 0.5, respectively. 19
2.4 Scatter plot of the fraction of calls wRTOTS estimated by the overflow model – left (resp. the A-Hypercube model – right) versus the fraction determined through simulation. The figures are percentages of the total rate of calls. 28
2.5 Differences between vehicle utilizations obtained from the overflow model and the simulation model (left). Corresponding differences for the A-Hypercube model. 32

3.1 Performance of ADP and the benchmark policy (solid thin line). 59
3.2 Performance of ADP with poorly chosen basis functions. 60
3.3 Performance of the 1,100 greedy policies obtained through random search. 62
3.4 Performance of ADP as a function of the frequency of the additional redeployments. 64
3.5 Performance of ADP for two values of κ and the benchmark policy (solid thin line). 67
3.6 Performance of ADP and benchmark policy for different call arrival rates. 70
3.7 Performance of ADP and benchmark policy for different fleet sizes. 71

4.1 First trial at iteratively solving the RLP 85
4.2 Iteratively solving the RLP with filtering constant $\theta = 0.01$ 87
4.3 Iteratively solving the RLP with filtering constant $\theta = 0.01$ and 100 simulation replications per iteration 88
4.4 Iteratively solving the RLP with filtering constant $\theta = 0.01$ and keeping constraints from most recent 5 iterations. 89
4.5 Iteratively solving the RLP with filtering constant $\theta = 10^{-5}$, and storing constraints from most recent 5 iterations. 90
4.6 Performance of best policy vs. frequency of extra-relocations . . 91

LIST OF ABBREVIATIONS

ADP	Approximate Dynamic Programming, see Chapter 3.
ALP	Approximate Linear Program, see Section 4.2.2.
API	Approximate Policy Iteration, see Chapter 3.3.
EMS	Emergency Medical Service, see Chapter 1.
LP	Linear program.
NDP	Neuro-dynamic Programming, see Chapter 1 syn. approximate dynamic programming.
RLP	Reduced Linear Program.
wRTOTS	with Response Times over the time standard (or threshold); see Section 1.1

CHAPTER 1

INTRODUCTION

> We do what we must, and call it
> by the best names.
>
> Ralph Waldo Emerson

The optimization of various aspects of emergency medical service (EMS) vehicle systems has been, since at least the mid 1960's, a very active area of research for applied mathematics and operations research. There have been hundreds of journal articles dealing with the development of models to support important decisions such us:

1. the locations, capacities and staffing of bases;
2. the scheduling of crews;
3. the number and type of vehicles to deploy at each base;
4. the choice of which vehicle to dispatch to an emergency; and
5. the redeployment of vehicles as a function of the system state.

There are several reasons why the design and operation of EMS systems has attracted so much attention from the operations research community. On one hand these issues are very important to society. Considering the large costs associated with obtaining and maintaining EMS equipment, and the highly qualified staff needed, it is of prime importance to make sure that available resources get the best possible use. On the other hand, the problems are rich and interesting from the mathematical point of view and require a high degree of ingenuity

from the researcher, both to keep up with the subtleties and complexities inherent to them as well as to come up with approaches that can be implemented in practice given limitations in data availability and computational resources.

This dissertation proposes new models for aspects 3 and 5 of ambulance logistics mentioned above. Specifically, we focus on the static allocation of an ambulance fleet to a given set of bases and on the real-time relocation of idle ambulances in an operating system. Both of these problems are closely related in that both deal with choosing optimal locations for ambulances as a function of the demand of the system. However, the former is strategic in character and allows for careful off-line computational procedures that deal with stationary properties of the system to be applied, whereas the latter requires the implementation of procedures that can be used in real-time and can react promptly to transitory changes in the system. Therefore, we treat each of these problems separately using fundamentally different analytical tools. The present work is thus naturally divided into two logically separated and self-contained parts which we now introduce in turn.

1.1 Static ambulance allocation

Emergency medical service providers face the problem of allocating a fixed number of ambulances among a set of bases. The ultimate goal is to ensure the best possible medical outcomes for patients. Though this is difficult to measure, an easily-measured proxy is the response time for a call, defined as the elapsed time from when a call is received to when an ambulance arrives at the scene. For studies demonstrating the relationship between response time and

probability of survival, see Erdogan, Erkut and Ingolfsson [21] and references therein. An industry standard is to attempt to ensure that a percentage p of all calls have response times under Δ minutes, with common choices being on the order of $p = 80\%$ and $\Delta = 10$. In this work, we will refer to Δ as the *time standard* or *time threshold*.

In Chapter 2, we provide two new computational models that can help with the ambulance allocation problem, where the goal is to minimize the fraction of calls with response times over the time standard (calls wRTOTS). We consider only *static* deployments, in the sense that ambulances return to their assigned bases whenever they are available to serve calls. This is in contrast to *dynamic* deployments, where ambulances are directed to different locations throughout their shifts in an attempt to better match anticipated demand. Dynamic deployments are the subject of Chapter 3.

Our first model in Chapter 2, which we call the *island* model, is *prescriptive* in nature, meaning that it implicitly searches over all possible allocations of ambulances to bases, and returns the one that yields the best performance, that is, the minimal percentage of calls wRTOTS. This ability to search over all possible allocations comes at a price, however. The island model uses a simplified objective function which, while related to the goal of minimizing the fraction of calls wRTOTS, is not an exact match for it. The quantity minimized is the fraction of calls that cannot be assigned to an ambulance immediately but are queued for service instead. The model also assumes that each base operates independently of all other bases, so that each base "is an island." The island model should therefore be viewed as a tool for obtaining insight into effective allocations, but its predictions should be refined through the use of more detailed methods.

Our second model, which we call the *overflow* model, can provide such refined predictions for a proposed allocation of ambulances to bases. As such, it is *descriptive* rather than *prescriptive* in that it provides performance predictions for a single proposed allocation. Computationally evaluating a single allocation can be done quickly (less than a second), and so it can be viewed as a tool for helping to quickly screen out poor ambulance allocations, and to identify potentially highly effective allocations for further investigation.

Both the island and overflow models are based on the Erlang loss formula, which is a standard result from queuing theory that has been used many times in EMS modeling, but not in the way we propose here.

The solution to the two-base version of the island model provides useful insights. In particular, it turns out that it is not optimal to allocate ambulances to bases in proportion to the call loads offered to bases. Instead, bases with lighter call loads should receive more than a proportional share of ambulances. This result can be seen as an economy of scale phenomenon. For a given utilization ratio (a ratio of demand offered to number of servers) bases with a larger number of ambulances – and serving a proportionally larger demand – offer a lower queueing ratio than bases with a small number of servers. Why is this important? This is essentially the problem faced by a central planner who must decide how many vehicles to allocate between two (or more) cities, where the two cities are far enough apart that vehicles based in one city cannot routinely attend calls in the other city. One might be tempted to allocate vehicles in proportion to the load, but our results show that this can be far from optimal. A similar situation arises in cities that, due to traffic congestion and/or geographic layout, can be broken down into 2 or more regions between which it is hard to

travel. This is precisely the case for the city of Auckland in New Zealand for example, where a major bridge linking the north and south portions of the city is virtually impassable during peak traffic periods. As the peak traffic periods approach, dispatchers might try to dispatch vehicles so as to appropriately distribute ambulances between the two regions.

The island model adds insight, but the assumption that bases do not interact is unfortunate if one wishes to apply the results to a typical city where interaction between bases is substantial. The overflow model relaxes this assumption and is, therefore, less tractable. The overflow model approximates, for a given ambulance allocation, the probability that a given base responds to calls originating in a given district of the city, for all base-district pairs. These probabilities can be used to compute the fraction of calls wRTOTS. This model can also be used to estimate other performance measures, as discussed later. The main idea is to use the Erlang loss formula with carefully chosen parameters to approximate the probability that all ambulances at one or more bases are busy, the so called busy probability. These probabilities are then used to compute dispatch probabilities, i.e., the probabilities that a call will be answered by an ambulance stationed at each base. The dispatch probabilities can be related back to the busy probabilities by means of a system of non-linear equations. Similar non-linear systems of equations have appeared many times before in the context of descriptive models for ambulance allocation and, in particular, in relation to the hypercube model Larson [43, 44], Jarvis [38], and its various extensions; see, e.g., Goldberg and Szidarovsky [30], Brandeau and Larson [13] and Budge, Ingolfsson and Erkut [15]. An important distinction of our work is that the earlier work formulates fixed point equations that involve a sophisticated version of the so-called Q factors, whereas our model avoids the need for Q factors by us-

ing the Erlang loss function. A similar use of the Erlang loss function to estimate overflow rates in multi-skill call centers appears in Koole and Talim [42]. The solution of the resulting system of equations is generally accomplished through iterative fixed point schemes. This is the approach we adopt in our work as well. Although we are not able to provide a proof of convergence of the resulting iterative scheme, the computational results suggest that there exists a unique fixed point and the fixed point can be computed extremely quickly. Therefore, as noted above, the overflow model can be used to pre-screen a large number of ambulance allocations, after which the best allocations can be studied through simulation.

Some of the insights developed in dealing with the static deployment problem proved helpful in attacking the more complicated problem of dynamic redeployment, which is the subject of the second part of the dissertation and we introduce next.

1.2 Dynamic programming based real time redeployment of ambulances

Ambulance redeployment is one possible approach that can help EMS managers cope with rising costs of medical equipment, increasing call volumes and worsening traffic conditions, and meet performance goals set by regulators and contracts, while at the same time taking better advantage of available resources. Ambulance redeployment, also known as relocation, move up, system-status management or dynamic repositioning, refers to any strategy by which a dispatcher repositions idle ambulances to compensate for others that are busy, and

hence, unavailable. The increasing availability of geographic information systems and the increasing affordability of computing power have finally created ideal conditions for bringing real-time ambulance redeployment approaches to fruitful implementation.

Chapter 3 presents the development and computational testing of an approach for making real-time ambulance redeployment decisions that is based on approximate dynamic programming (ADP). We begin by formulating the ambulance redeployment problem as a stochastic dynamic program (SDP). This kind of formulation allows our model to capture the random evolution of the system over time. The expected (discounted) number of calls with response times over the time standard plays the role of the cost-to-go function which is to be minimized. The stochastic dynamic program involves a system evolving in continuous time on a high-dimensional and uncountable state space and, due to its high complexity, is intractable by means of exact DP techniques. ADP is then adopted as a way to circumvent this difficulty. The most common and best understood approach to ADP, which we also adopt in our work, starts by approximating the arbitrary value functions by linear combinations of a small set of basis functions of the state, the so-called *feature functions*. The features functions have to be especially and carefully tailored to the particular application in order to guarantee the effectiveness of the scheme. Once a sensible set of feature functions is chosen, the problem becomes that of the tuning the coefficients in the linear combination to best approximate the actual optimal cost-to-go function. There are several ways to achieve this.

The ADP algorithm termed Approximate Policy Iteration (API), which is studied in Chapter 3, tunes the coefficients of the value function approximation

through an iterative simulation-based method. Each iteration of the API algorithm consists of two steps. In the first step, we simulate the trajectory of the greedy policy induced by the current value function approximation and collect cost trajectories of the system. This is the analog of policy evaluation in the exact version of the policy iteration algorithm. In the second step, we tune the parameters of the value function approximation by solving a regression problem that fits the value function approximation to the collected cost trajectories. This is the analog of policy improvement in the exact algorithm. The latter step yields a new set of parameters that characterize new value function approximations, and so, we can go back and repeat the same two steps.

We emphasize here that what allows the successful application of the general ADP framework to a problem as complex as real-time ambulance redeployment is the development of a refined approximation architecture, i.e., a set of basis functions $\{\phi_p(\cdot) : p = 1, \ldots, P\}$, that is able to capture the essence of how an EMS system configuration at a given time determines future performance. A combination of a great deal of experimentation and use of the insights gained from the first part of this work proved very helpful in achieving this. We consider the development of this architecture a major contribution of our work, as this development took literally months of exhaustive exploration and experimentation of many different alternative function architectures.

In the second part of Chapter 3, we provide computational experiments on EMS systems in two scenarios: the city of Edmonton, Alberta Canada and another much bigger city. Our results indicate that ADP has the potential to obtain high-quality redeployment policies in real systems. They also show that our approach compares favorably with benchmark policies that are similar to those

used in practice.

Chapter 4 describes the implementation of a linear-programming based ADP approach that approximately solves the dynamic program by constructing and solving what is termed the reduced linear program (RLP). To the best of our knowledge, the application described in this chapter is the first application of LP-based ADP techniques to a problem involving an uncountable state-space and continuous-time event-based evolution. The constraints in the RLP come from state-action pairs visited during the course of simulation under some policy. The decision variables in the RLP are the tunable parameters of the linear approximation architecture. Thus, after the simulation is finished and the linear program is solved, the solution vector will determine a value function approximation. The greedy policy with respect to this value function will hopefully be better than the starting one. This approach lends itself naturally to be implemented as part of a larger iteratively scheme, in which new simulations are carried out and a new set of constraints are sampled under the new policy, much in the same way as the API from Chapter 3.

We carry out numerical tests on this iterative approach for the city of Edmonton scenario studied in Chapter 3. Some difficulties related to the ill-conditioning of the RLPs generated by the algorithm are found and resolved through refinements in the way constraints are collected and on the way the tunable parameters are updated from one iteration to the next. We also find that, in the cases considered, the best policies generated by the algorithm have practically the same performance as the best policies obtained by the API approach of Chapter 3.

CHAPTER 2

ERLANG LOSS MODELS FOR THE STATIC DEPLOYMENT OF AMBULANCES

> Times have not become more violent. They have just become more televised.
>
> Marilyn Manson

How should one allocate a fleet of ambulances to fixed bases with the goal of keeping response times to calls as small as possible? We present two new models for this problem, both of which are based on the Erlang loss formula.

The first model is stylized and mostly intended to provided managerial insight into the ambulance deployment problem. The model is an integer program that allocates a fixed number of ambulance among a set of bases. The objective is to minimize the number of calls that do not find an available ambulance when they are received, and we capture this fraction by using the Erlang loss function. Among other things, it shows that allocating ambulances in proportion to the offered load is not necessarily optimal. Instead, bases with lighter call loads receive more than a proportional share of the ambulances. The main simplifying assumption of this model is that it does not allow collaboration between different bases, or put in a slightly different way, this model assumes a priori knowledge of the amounts of demand served by the different bases. Of course, bases always interact to some extent, but there are situations where bases cannot interact as much as one would like owing to natural barriers such as the sea harbors in Auckland, New Zealand or traffic congestion on arterial roadways.

The results of our first model should also be useful for a centralized planner who is responsible for allocating the EMS assets between different cities that do not share resources operationally.

The first model adds insight, but the assumption that the bases do not interact is unfit. Our second model relaxes this assumption, though at the expense of loss of tractability. The second model estimates, for a given ambulance allocation, the fraction of calls whose response times are longer than some time standard. It can be used to screen potential allocations to identify top candidates for further investigation. Computational experiments provide useful insights. We carry out a careful comparison of the second model with the A-hypercube model which shows that our model has comparable and in many cases better accuracy measures. Thus our model can be used as a pre-screening tool to, in combination with detailed simulation, efficiently identify nearly optimal static ambulance allocations.

This chapter is organized as follows. Section 2.1 presents a a review of previous literature on the static allocation problem. Section 2.2 introduces the island model, and we study the two-base version of the island model in detail in Section 2.3. Section 2.4 describes the overflow model, and Section 2.5 compares performance estimates obtained via the overflow model, the Approximate Hypercube model and a simulation model. We conclude in Section 2.6.

2.1 Literature Review

Static ambulance deployment problems have received a great deal of attention in the literature. We present a brief survey here to give a feel for the primary

approaches studied in the past. For more detailed surveys, we recommend the excellent reviews by Swersey [56], Brotcorne, Laporte and Semet [14], Goldberg [27] and Green and Kolesar [31].

As mentioned in the introduction, prescriptive methods are optimization oriented in that they search over all possible ambulance allocations to identify promising allocations, and they typically use simplified performance measures to allow for efficient search procedures. Important early work on prescriptive models includes Toregas, Saqin, ReVelle and Berman [58] and Church and ReVelle [16]. The later model by Daskin [17] attempts to capture some of the stochastic aspects of the problem under the assumption that the ambulances are statistically independent. Batta, Dolan and Krishnamurthy [6] build on this model to relax the independence assumption, whereas ReVelle and Hogan [50] and Marianov and ReVelle [46] extend the earlier models by adding constraints on the minimum number of ambulances required to cover various zones. The last two papers are related to our work in the sense that the minimum number of ambulances are obtained from the Erlang loss formula. More recently, Erdogan et al. [21] incorporate medical outcomes into the objective function by concentrating on the lengths of the response times rather than the fraction of calls with response times below a certain time standard. All of the papers mentioned in this paragraph thus far generally use linear integer programming formulations, but nonlinear models have also been used by several authors. For example, Goldberg and Paz [29] embed a heuristic search procedure into the approximate hypercube model proposed by Larson [44]. The underlying theme of this work is to use tractable approximations for the performance of the system to guide a search procedure.

Descriptive methods, on the other hand, do not explicitly optimize over a feasible set of allocation. Instead, they focus on evaluating single allocations in order to provide more accurate performance predictions than are possible with the simplified objective functions used in prescriptive models. This area is dominated by the hypercube method Larson [43] and its extensions, which include Larson [44], Jarvis [38], Jarvis [39], Berman and Larson [10], Brandeau and Larson [13], Goldberg and Szidarovsky [30] and Budge et al. [15]. The descriptive model presented in Section 2.4 falls into this line of development in that it formulates a non-linear system of equations relating zone demands to ambulance busy probabilities. The main difference is that earlier work formulates equations that involve a sophisticated version of the so called Q factors, whereas our model completely avoids the need for Q factors by using the Erlang loss function. Another recent work, also based on queuing theory, is Singer and Donoso [54]. Another important descriptive approach is, of course, simulation. Simulation has been used in a large portion of ambulance deployment studies either to analyze the performance of other models or as a tool in and of itself. For an overview, see Henderson and Mason [34].

2.2 The Island Model

In this section we present a prescriptive model (the island model) for the static ambulance deployment problem. We have N ambulances to allocate among a set of bases. The set of bases is $\mathcal{B}$ and we can allocate at most c_b ambulances to base b. An ambulance deployed at base b can serve calls at rate μ_b and the calls that are offered to base b arrive according to a Poisson process with rate λ_b.

Our goal here is to obtain insight about the problem rather than to obtain predictions that could be used immediately in real organizations. Accordingly, we adopt a model that is mathematically tractable at the expense of some realism. In the island model, each base operates independently. Calls that arrive for a base when all ambulances that are located there are busy are lost, meaning that they are assumed handled by some outside agency (e.g., the fire department, or a competing EMS organization). So in the island model bases do not handle the overflow calls from other bases, and $\{\lambda_b : b \in \mathcal{B}\}$ are known parameters that do not depend on the ambulance allocation. We relax this strong assumption later in Section 2.4.

If we allocate n_b ambulances to base b, then each base operates as an $M/G/n_b/n_b$ queue, i.e. a queue with exponential interarrival times, general service time distribution, n_b servers and a maximum of n_b customers in the queue. This implies that we can compute the probability that an arriving call offered to base b finds all n_b ambulances busy by using the Erlang loss formula

$$\mathcal{E}(n_b, \lambda_b/\mu_b) = \frac{[\lambda_b/\mu_b]^{n_b}/n_b!}{\sum_{j=0}^{n_b}[\lambda_b/\mu_b]^j/j!}.$$

See, for instance, Gross and Harris [32] for proof that this formula is valid for general service time distributions.

In this case, we can solve the problem

$$\min \quad \sum_{b\in\mathcal{B}} \lambda_b \, \mathcal{E}(n_b, \lambda_b/\mu_b) \tag{2.1}$$

$$\text{subject to} \quad \sum_{b\in\mathcal{B}} n_b = N \tag{2.2}$$

$$0 \le n_b \le c_b \qquad b \in \mathcal{B} \tag{2.3}$$

$$n_b \text{ integer} \qquad b \in \mathcal{B} \tag{2.4}$$

to decide how many ambulances should be allocated to each base. Each term in the objective function, $\lambda_b \mathcal{E}(n_b, \lambda_b/\mu_b)$, corresponds to the expected number of lost calls offered to base b per unit time. As shown, for example, in Harel [33], $\mathcal{E}(n_b, \lambda_b/\mu_b)$ is a convex function of n_b, which implies that the objective function of Problem (2.1)-(2.4) is a separable convex function. Consequently, Problem (2.1)-(2.4) can be solved efficiently through simple marginal analysis; see Fox [24]. (One takes a greedy approach, adding ambulances one at a time, respecting the base capacities but otherwise choosing the base that leads to the greatest reduction in the objective function.) In certain situations, such as the ones considered in the next section, and in order to get a better qualitative understanding of this model, it will prove useful to relax the integrality requirement in (2.4) and use continuous version of the Erlang loss formula [37].

Recall that, in this model, calls that arrive when all ambulances at a base are busy are "lost." The proportion of such calls is small when the load on a base is small. Hence, it is in this lightly loaded regime where the model is most appropriate.

The queueing model underlying the Erlang loss function assumes that the service times (including travel time, time at scene, etc.) for successive calls are independent. This assumption does not hold precisely, because the locations of ambulances are affected by the locations of previous calls. The assumption is reasonable when a large portion of calls are served by ambulances that are already waiting at the base. As reported in Richards [51], even in systems having utilizations of around 35%, the fraction of calls responded to from the road is usually only around 10%. Furthermore, the time spent by an ambulance at the scene typically dominates the travel time [15], so the dependence between calls,

even when calls are often responded to from the road, is mild.

The most unfortunate assumption in Problem (2.1)-(2.4) is that we have *a priori* knowledge of what portions of the demand are served by the different bases. In reality, the total rate of call arrivals into the system is known, but the proportions of the calls served by the different bases emerge as a result of the ambulance dispatch policy. Problem (2.1)-(2.4) assumes that fixed and known portions of the calls arriving into the system are offered to the different bases and the bases do not help each other even if they are not able to cope with their offered call loads.

2.3 Insights from the Island Model

Consider Problem (2.1)-(2.4) in the case where we have two bases, so that $\mathcal{B} = \{1, 2\}$. Our goal is to obtain insight into the optimal allocation of ambulances to bases. As we will see, allocating ambulances in proportion to the call loads at the bases can be highly suboptimal, but as the total load increases, this sub-optimality decreases.

We first let $(\mu_1, \mu_2) = (1, 1)$ and $N = 20$, and solve Problem (2.1)-(2.4) for different values of (λ_1, λ_2). We choose $(\mu_1, \mu_2) = (1, 1)$ only for convenience and other service rates can be captured by simply scaling our results. The regions and the associated labels in Figure 2.1 show the optimal solutions to Problem (2.1)-(2.4) for different values of (λ_1, λ_2). For example, if $(\lambda_1, \lambda_2) = (3, 2)$, then the optimal solution to Problem (2.1)-(2.4) is $(n_1, n_2) = (11, 9)$. In Figure 2.1, the shaded area covers those regions for which allocating ambulances in proportion to call loads differs from the optimal allocation. It is easy to see that allocating

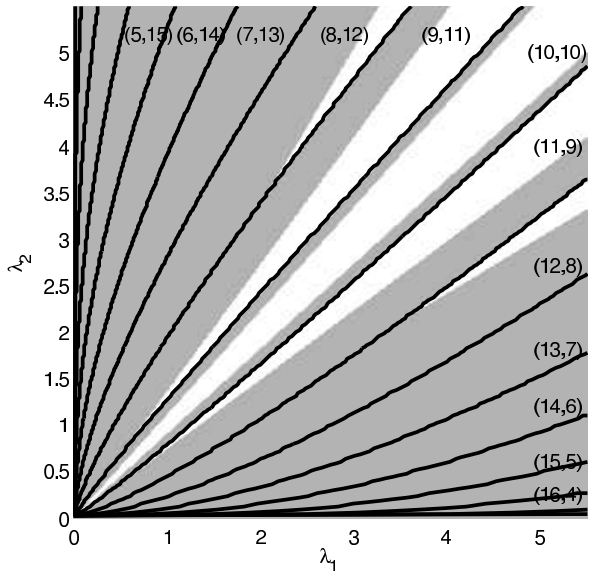

Figure 2.1: Optimal solutions to Problem (2.1)-(2.4) for different values of (λ_1, λ_2).

ambulances in proportion to the call loads offered to the different bases is almost never optimal for low utilization regimes and it is only optimal for the higher utilization regimes if λ_1 is close to λ_2. For example, if $(\lambda_1, \lambda_2) = (5, 2.5)$, then we have $\lambda_1/\lambda_2 = 2$, but the optimal solution to Problem (2.1)-(2.4) is $(n_1, n_2) = (12, 8)$ and $n_1/n_2 = 1.5$.

Exploring further, suppose that $(\mu_1, \mu_2) = (1, 1)$ and $N = 10$. Since the service rates are equal to 1, the offered load per server is $\rho = (\lambda_1 + \lambda_2)/N$. We vary λ_1 but keep ρ and N fixed. In Figure 2.2, we plot n_1^*, the optimal number of ambulances assigned to Base 1 as a function of λ_1. We do this for 3 different values of ρ. In each case, the units of the x-axis have been linearly scaled and correspond to $\lambda_1/(\lambda_1 + \lambda_2)$ which is the fraction of the total demand $\lambda_1 + \lambda_2$ that is offered to the first base. We note that Figure 2.2 uses a continuous version of the Erlang loss formula to allow fractional ambulance allocations and this allows us to produce

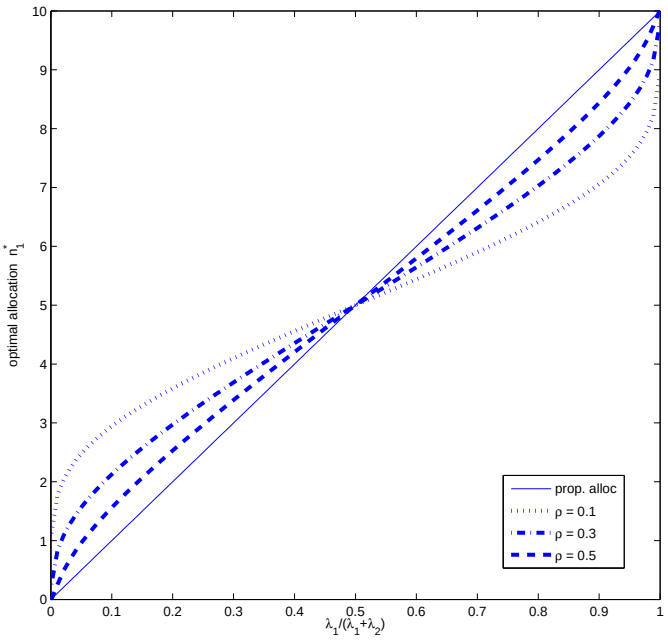

Figure 2.2: Value of n_1 in the optimal solutions $(n_1, N-n_1)$ to Problem (2.1)-(2.4) for different values of $\lambda_1/(\lambda_1 + \lambda_2)$.

smooth plots [37].

If proportional allocation were optimal, we would have $n_1/N = \lambda_1/(\lambda_1 + \lambda_2)$ and the optimal solutions to Problem (2.1)-(2.4) as a function of $\lambda_1/(\lambda_1 + \lambda_2)$ would lie on the straight line in Figure 2.2, for any value of the offered load per server. However, the figure shows that optimal solutions differ substantially from proportional allocation, especially when the offered load per server is low. When the offered load per server is high, proportional allocation is close to optimal. However, this requires offered load per server values as large as 0.5, whereas a more typical value in practice is 0.3; see, e.g., Budge et al. [15]. When $\lambda_1/(\lambda_1 + \lambda_2) < 0.5$, the number of ambulances allocated to the first base is larger than that suggested by proportional allocation. In other words, bases with lighter loads receive more than their proportional share of ambulances.

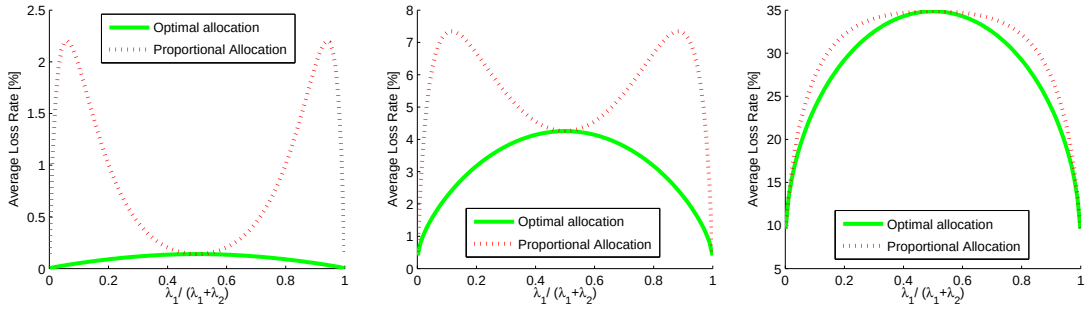

Figure 2.3: Rate of lost calls for proportional allocation and the optimal allocation, for $\rho = 0.15, 0.3$ and 0.5, respectively.

The intuitive reason for this behaviour is that, for higher call rates the number of calls wRTOTS becomes proportional to the utilization, regardless of the number of ambulances thus proportional allocation ensures that different bases have roughly the same loss. On the other hand, at lower utilization rates bigger capacity bases are significantly more robust and provide a much

But how significant is the sub-optimality of proportional allocation?

Figure 2.3 gives the loss rates for proportional allocation and the optimal allocation as a function of $\lambda_1/(\lambda_1 + \lambda_2)$. The results indicate that the cost of using proportional allocation can be very large. For example, if $\lambda_1/(\lambda_1 + \lambda_2) = 0.1$, then proportional allocation loses about 7% of the calls, whereas the optimal solution loses about 2% of the calls. This is certainly a practically significant difference. Similar to Figure 2.2, Figure 2.3 uses a continuous version of the Erlang loss formula.

The non-optimality of proportional allocation appears to be a general phenomenon, occurring in situations with more than 2 bases. Consider a situation with 6 ambulances and 3 bases with offered demands $(\lambda_1, \lambda_2, \lambda_3) = (1.2, 0.3, 0.3)$, respectively, so that $\rho = 0.3$. Proportional allocation yields $(n_1, n_2, n_3) = (4, 1, 1)$

which has a loss rate of 9.4% of calls, whereas $(n_1, n_2, n_3) = (3, 2, 2)$ has a loss rate of 7.1% of calls. Similar examples are easily constructed for greater numbers of ambulance and bases. To see why this observation should hold for more than 2 bases, consider the (unknown) optimal allocation of ambulances to bases. Select any 2 bases. The allocation of ambulances to these 2 bases must be optimal for a 2-base system, so our observation above should hold in this case, and the lower-loaded base receives a more-than proportional number of the ambulances shared by the 2 bases.

2.4 The Overflow Model

In this section we propose a descriptive model (the overflow model) that estimates performance measures for a given ambulance allocation. The overflow model relaxes some of the critical assumptions of the island model in Section 2.2. In particular, it does not require that we know the call loads offered to different bases, and allows bases to cooperate.

The primary use we have in mind for the overflow model is to quickly evaluate a large number of possible ambulance allocations. We can then evaluate a small set of the most promising allocations through a detailed simulation study. Thus the overflow model is not meant to replace simulation but rather to complement it, by enabling pre-screening and fast discarding of many possible allocations.

We start by considering a simple extension of the island model meant to overcome the assumption that $\{\lambda_b : b \in \mathcal{B}\}$ are known parameters. One can attempt to do this by turning these parameters into decision variables in Problem

(2.1)-(2.4). This idea amounts to solving a problem of the form

$$\begin{aligned}
\min \quad & \sum_{b\in\mathcal{B}} \lambda_b \, \mathcal{E}(n_b, \lambda_b/\mu_b) \\
\text{subject to} \quad & \sum_{b\in\mathcal{B}} n_b = N \\
& 0 \le n_b \le c_b \qquad b \in \mathcal{B} \\
& n_b \text{ integer} \qquad b \in \mathcal{B} \\
& \sum_{b\in\mathcal{B}} \lambda_b = \lambda \\
& \lambda_b \ge 0 \qquad b \in \mathcal{B},
\end{aligned}$$

where λ is the total rate of call arrivals into the system, and $\{n_b : b \in \mathcal{B}\}$ and $\{\lambda_b : b \in \mathcal{B}\}$ are the decision variables. This problem finds an allocation of the calls and ambulances among the bases to minimize the number of lost calls, and it mimics the natural load balancing sought after by dispatchers in practice. Unfortunately, numerical evaluations reveal that $\lambda_b \, \mathcal{E}(n_b, \lambda_b/\mu_b)$ is not jointly convex in (n_b, λ_b). Therefore, there are no guarantees that the new problem has a unique local minimum, and finding a global optimum may be almost as difficult as enumerating all possible values of $\{n_b : b \in \mathcal{B}\}$ that satisfy Constraints (2.2)-(2.4).

In view of this we consider a different approach that does not directly take optimizing the expected Erlang loss as the criterion for dividing the demands but instead views the problem as a queueing system with a spatial structure.

We start by assuming that there are a number of demand nodes indexed by $j = 1, \ldots, J$. Each demand node j generates a Poisson arrival process of calls with rate d_j and the processes corresponding to different demand points are independent. The set of bases is $\mathcal{B} = \{1, \ldots, B\}$.

Each demand node j has a fixed priority ranking of all bases $j(1), \ldots, j(B)$,

which might correspond, for instance, to how far each base is from node j. A call originating at node j is first offered to an ambulance based at $j(1)$, and if no ambulance is available there, then it is offered to an ambulance based at $j(2)$, and so forth. If no ambulance is available at any base, then the call is lost and assumed to be answered by some alternative agent. The term "offered" emphasizes that some calls contribute to the load on a base, even if they are not actually served by that base. We say that a base is busy if all ambulances stationed there are busy.

Let s_{jk} denote the probability, under stationarity, that a call originating at node j will be answered by an ambulance from base $j(k)$. We extend the definition of s_{jk} to include $k = B + 1$, so that $s_{j,B+1}$ is the probability that no ambulance is available.

Remark 1 *The probability that all ambulances are busy is typically extremely small. Indeed, if each ambulance is independently busy with probability ρ, where ρ is the overall utilization, then $s_{j,B+1} = \rho^N$. This value is negligible for systems with modest utilization and many ambulances. In general the true probability that all ambulances are busy is larger than this approximation, but still practically negligible, except in cases of extreme load that do not typically arise in daily operations.*

Suppose we are able to obtain (approximations for) these probabilities. Then it is easy to compute (approximations for) any performance measure relating to the response time to calls, for which the conditional expected value given the originating demand point j and responding base k can be (approximated or) computed. Let ψ_{jk} denote this conditional expectation. Then the conditional

expectation $\psi(j)$ given the originating demand point j is

$$\psi(j) := \sum_{k=1}^{B} s_{jk}\psi_{jk}. \tag{2.5}$$

Moreover, ψ is then given by

$$\psi = \frac{\sum_{j=1}^{J} d_j\psi(j)}{\sum_{j=1}^{J} d_j}. \tag{2.6}$$

Examples of such performance measures include the probability of reaching a call within a given time threshold, the expected response time and, as considered in Erdogan et al. [21], the survival probability.

In our analysis, we focus on the probability of reaching a call within a given time threshold, which we denote by r. Assuming deterministic travel times, if demand node j can be reached from, and only from, the first $N(j)$ bases in demand node j's list within the time standard, the specialization of equation (2.5) to this measure simplifies to

$$r(j) = \sum_{k=1}^{N(j)} s_{jk}. \tag{2.7}$$

In this setup, it is possible to derive a system of equations relating the probabilities s_{jk} and the node demands that has the form

$$S = f(S), \tag{2.8}$$

where S stands for the $J \times (B+1)$ matrix $[S]_{jk} = s_{jk}$ and f is a certain non-linear function.

We present the detailed derivation in Appendix A and only present the most important points here. First, the total demand offered to base b is the sum of a primary demand, coming from nodes for which b is the preferred base, plus an

overflow demand coming from other nodes, when all bases preferred to b by those nodes are busy. The probability that all ambulances at a particular base are busy can be estimated from the demand offered to it using the Erlang loss formula. However, when trying to estimate the probability that a call from node j finds all servers at its k-th preferred base busy, conditioned on all ambulances in the previous $k-1$ bases to be busy, one has to take into account the addition to the demand offered to base k from overflow demand from the first $k-1$ bases. Careful estimation of the new demands resulting from this conditioning leads to a system of non-linear equations for the matrix of probabilities s_{jk} which can be put in the form (2.8).

2.5 Computational Results for the Overflow Model, the A-Hypercube Model and Simulation

In this section, we present a case study based on data from the Edmonton, Alberta, EMS. The key performance measure considered is the percentage of calls wRTOTS, although we also look at vehicle utilization. We apply both the overflow model introduced in the previous section and the approximate hypercube (A-Hypercube) model developed in Larson [44]. We use the second method in Section 4 of Larson [44] to normalize the ambulance dispatch probabilities of the A-Hypercube model. Our goal is to verify that the performance measure estimates computed through the overflow model are accurate enough to reliably assist in identifying high-quality ambulance allocations for further exploration.

Some important structural differences between the A-Hypercube approximation scheme and the overflow model are worth mentioning here. In the

A-Hypercube model, the fundamental descriptive variables are the individual vehicle utilizations $\rho = (\rho_a, a = 1, \ldots, N)$, where N is the total number of ambulances. In the overflow model, the fundamental quantities are the probabilities s_{jk} that a call from any give zone is served by any given base. As in our model, the A-Hypercube estimation of ρ is based upon the solution of an equation of the form $\rho = f_H(\rho)$, where f_H is a certain non-linear function. In the original procedure, proposed by Larson [44], this is done by initializing $\rho^0 = (\overline{\rho}, \overline{\rho}, \ldots \overline{\rho})$, where $\overline{\rho} := \lambda/\mu N$ is the average global utilization of the system, and producing a sequence of estimates $\rho^1, \rho^2, \ldots$, as follows. At each step, one first lets $\tilde{\rho}^k = f_H(\rho^{k-1})$ and then computes $\rho^k = \overline{\rho} \cdot (\tilde{\rho}^k)/(\sum_{a=1}^{N} \tilde{\rho}_a^k)$, thereby re-normalizing the utilizations to the average $\overline{\rho}$. The procedure we use to solve the fixed point equation (2.8) generates a sequence of matrices $S^0, S^1, \ldots$ in an analogous way, but it does not enforce any normalization on S. The natural normalization condition, namely, that $\sum_k s_{jk} = 1$, for every j is automatically preserved by our f. See Appendix B for a discussion on ways to initialize the matrix S.

The A-Hypercube iterative procedure stops when $\|\rho^{k+1} - \rho^k\|_1 < \epsilon$, a given tolerance. In order to establish a fair comparison between the speeds of convergence for both methods, we set the stopping condition for the overflow model to be $\|R(S^{k+1}) - R(S^k)\|_1 < \epsilon$, where $R(S)$ is a function that computes individual vehicle utilizations from the probability matrix S. An exact description of this function can be found in Appendix C.

Our input data are taken from Budge et al. [15], describing the ambulance system operating in Edmonton, Alberta. The system is medium sized with 16 bases and 180 demand nodes. We experimented with allocations of 12 and 16 ambulances and demand rates ranging between 2 and 7 calls per hour. The

service rate is about $\mu = 0.97$ calls per hour per ambulance, which corresponds to an average service time of about 62 minutes per call.

A computation shows that there are approximately 400,000 different ways of allocating 12 ambulances among 16 bases given the capacity constraints for the bases in this system. The number of allocations of 16 ambulances is also very close to 400,000. In our experimental setup, we select subsets of 1,000 allocations among the 400,000 possible ones, for each of these two scenarios. We apply both the overflow model and the A-Hypercube model to each of these allocations to estimate the base (resp. vehicle) utilizations. From these we calculate predictions for the system-wide percentage of calls wRTOTS as

$$100\% - \frac{\sum_{j=1}^{J} d_j r(j)}{\sum_{j=1}^{J} d_j},$$

where $r(j)$ is given by (2.7), and d_j is the demand rate at location j.

To provide a benchmark for comparison, we use discrete-event simulation to compute the same performance measure for each of the 1,000 ambulance allocations in a simplified model of ambulance operations. The specifics of the simulation model are as follows. Calls are generated in each of the 180 locations as independent Poisson processes with constant rates equal to the historical demand rates. All calls are assumed urgent. The ambulance dispatching policy is to send the closest available ambulance to a call, and if there are no available ambulances, then the call is put in a first-in-first-out queue. (Putting a call in a queue almost always implies that the call will not meet the time standard.) In the simulation, travel times between each base and demand zone are deterministic and taken to be equal to the average travel time observed from historical data. Ambulances spend an exponentially distributed amount of time at the scene with a mean of 12 minutes. Patients require transport to a hospital with

probability 0.75. Given that a patient is to be transported to a hospital, the choice of hospital is independent of all else and chosen according to historical probabilities. The time spent at the hospital is Weibull distributed with a mean of 30.4 minutes and a shape parameter of 2.5.

For each allocation, we simulate the system for two weeks and make 50 runs to estimate the fraction of calls wRTOTS. Fifty replications are enough to estimate this fraction with a 95% confidence interval of width 0.6%. More importantly, since we are using common random numbers (i.e., the same set of calls) for all allocations, 50 replications is enough to establish a difference between allocations whose mean performances differ by 0.5% or more. For instance, in the experiment with 12 ambulances and $\lambda = 3$ calls per hour, the simulation of allocations #345 and #435 yields mean costs of 16.7% ± 0.3% and 16.5% ± 0.3%, respectively, and a difference of 0.3% ± 0.3%. Thus, the second allocation is better than the first one with high confidence, even when the confidence intervals for their mean costs have a big overlap. The average response time is, of course, a function of the allocation. For the best allocation in the situation above, the average response time is 5.8 min. The average time of transport to the hospital is observed to be 16.4 min, which is considerably larger due to the fact that there are significantly fewer hospitals than bases and that the ambulance travels at regular speed when driving to the hospital.

The chart on the left-hand side of Figure 2.4 shows a scatter plot of the estimate of the fraction of calls wRTOTS as predicted by the overflow model versus the corresponding fraction measured from the simulation model for 1,000 ambulance allocations. The total number of ambulances is 12, and the total call rate is $\lambda = 3$ calls/hr which corresponds to an average utilization $\bar{\rho} = 0.26$. The chart

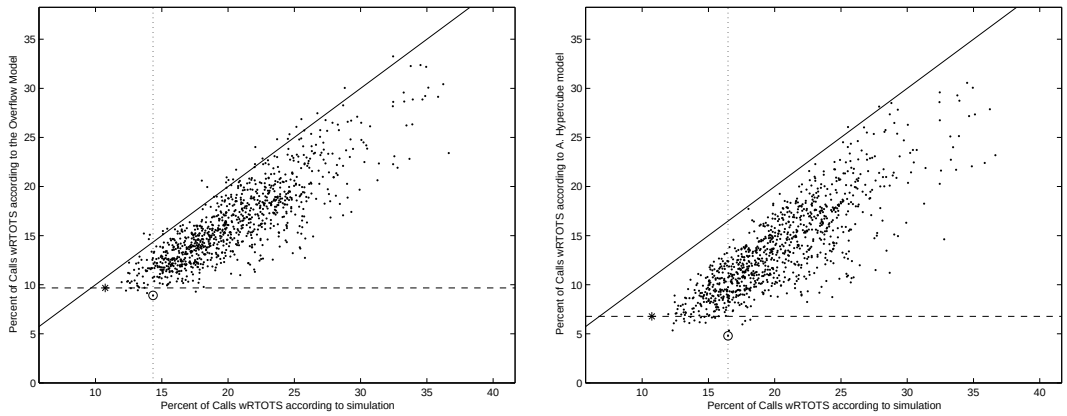

Figure 2.4: Scatter plot of the fraction of calls wRTOTS estimated by the overflow model – left (resp. the A-Hypercube model – right) versus the fraction determined through simulation. The figures are percentages of the total rate of calls.

on the right-hand side of the same figure shows the scatter plot of estimates yielded by the A-Hypercube model versus those from the simulation model, for the same 1,000 allocations. The solid line on both charts is the line $y = x$ on which the points would lie if the predictions of the approximate models agreed perfectly with the simulation measurements.

The overflow model predicts the fraction of calls wRTOTS more accurately than the A-Hypercube model. The predictions of both models exhibit a negative bias, with the A-Hypercube model bias being more pronounced. The root mean-squared error (RMSE) of the overflow model predictions is 4.8% whereas the RMSE of the A-Hypercube model predictions was 7.2%. The Spearman's (rank) correlation between the overflow model's predictions and the simulation measurements is 0.86, which is slightly better than that for the A-Hypercube model (0.82).

As stated earlier, the primary purpose of the overflow model is to serve as

a pre-screening tool to evaluate a large set of possible allocations quickly in order to identify the most promising ones to evaluate using detailed simulation. To investigate the effectiveness of this approach we define F as the fraction of allocations that have a model-predicted cost below that of the allocation having minimum simulated cost. The usefulness of measure F is that it is proportional to the number of allocations one has to evaluate through simulation in increasing order of model-evaluated cost before one evaluates the optimal allocation. In the charts in Figure 2.4 the allocation having minimum simulated cost is marked with a $*$ and the fraction just defined corresponds to the fraction of allocations below the horizontal dashed line. For the overflow model we have $F = 0.006$. For the A-Hypercube model we have $F = 0.020$, which is considerably larger.

The results we have just described are not uncommon in the low to medium-low utilization range. Table 2.1 summarizes the results of a series of experiments carried out on allocations having 12 and 16 ambulances over a range of system utilizations. In general we see that the overflow model has greater accuracy in predicting the percentage of calls wRTOTS, as evidenced by lower value of the RMSEs throughout. Rank correlations (R. Corr.) are similar for both models, with the A-Hypercube being significantly better only for the highest utilizations. The overflow model F-fractions are comparatively lower than those of the A-Hypercube for the lowest utilizations but higher for the highest utilizations.

Another measure registered in Table 2.1, of somewhat lesser importance than F, is G, defined as the fraction of allocations having simulated cost lower than that of the best model-ranked policy. In the charts, this corresponds to the fraction of points to the left of the dotted line. This fraction is a measure of how

Table 2.1: Comparison of accuracy measures of the overflow and A-Hypercube models for a range of average utilizations. An $*$ indicates an instance where a measure for a model is significantly better than the same measure for the other model.

Sim. Params.			Overflow Model				A-Hypercube Model				
N	λ	$\bar{\rho}$	R.Corr	RMSE	F	G	R.Corr	RMSE	F	G	O.D.
12	2	0.17	0.85	*5.8%	*0.006	0.004	0.83	6.6%	0.025	0.004	0
16	3	0.19	*0.86	*4.2%	0.001	*0.003	0.82	5.1%	0.002	0.018	1
12	3	0.26	0.84	*4.8%	*0.006	*0.046	0.82	7.2%	0.020	0.190	0
16	5	0.32	0.73	*2.5%	0.090	0.261	0.74	6.7%	*0.020	0.301	3
12	5	0.43	0.71	*4.0%	0.122	0.060	0.79	7.8%	*0.014	0.052	2
16	7	0.45	0.49	*6.5%	0.457	* 0.035	*0.67	7.5%	*0.033	0.191	0

far the best model-ranked policy is from optimality. A notion of the magnitude of this quantity could be helpful for ambulance managers who do not have the means to evaluate allocations through detailed simulation but only have a model upon which to rely. The overflow model yields G-ratios that are at least as low as those yielded by the A-Hypercube model in 5 of the six cases presented in Table 2.1 and yields significantly lower values in 3 of them.

Another important measure of the quality of these methods is how well they predict the utilization of individual vehicles. The chart on the left-hand side of Figure 2.5 shows a histogram of the absolute difference between the individual vehicle utilizations predicted by the overflow model and those obtained by simulation. The chart on the right does the same for the A-Hypercube model predicted utilizations. The differences are taken individually for each of the vehicles and for each of the 1,000 evaluated allocations. A positive difference indicates that the model-predicted utilization is larger than the simulated one.

Table 2.2: Error statistics for individual vehicle utilizations according to the two models.

Sim.Params		Overflow Model		A-Hypercube Model	
N	λ	Error	Std.	Error	Std.
16	3	1.3%	2.5%	0.7 %	4.0%
12	3	2.5%	2.9%	0.6 %	4.8%
16	5	5.4%	3.6%	1.0 %	5.6%
12	5	9.6%	3.9%	0.2 %	5.8%
16	7	13.3%	4.5%	0.9 %	6.6%

The utilization differences for the overflow model have a mean of 2.5% and a standard deviation of 2.9% whereas those of the A-Hypercube model are less biased, with a mean of 0.6%, but less precise, with a standard deviation of 4.8%. The situation exemplified here is not atypical. As Table 2.2 shows, for the same set of scenarios as before, overflow-model predicted utilizations have a positive systematic error that grows as the system utilization grows but smaller standard deviations than those yielded by A-Hypercube model predictions. On the other hand, the A-Hypercube predictions have a very small bias. This is not surprising considering that, as mentioned earlier, the A-Hypercube procedure renormalizes base utilizations to the system utilization at each step.

We briefly comment now on the computational speed of both methods. The fixed point procedures were implemented in MATLAB and run on a 3.2 GHz Pentium IV PC running Linux. For the situation with 12 ambulances and λ = 3 calls/hr, each iteration of the A-Hypercube solution method takes 7.8 ± 2.5 ms whereas each iteration of the overflow method took 240 ± 65 ms. Thus,

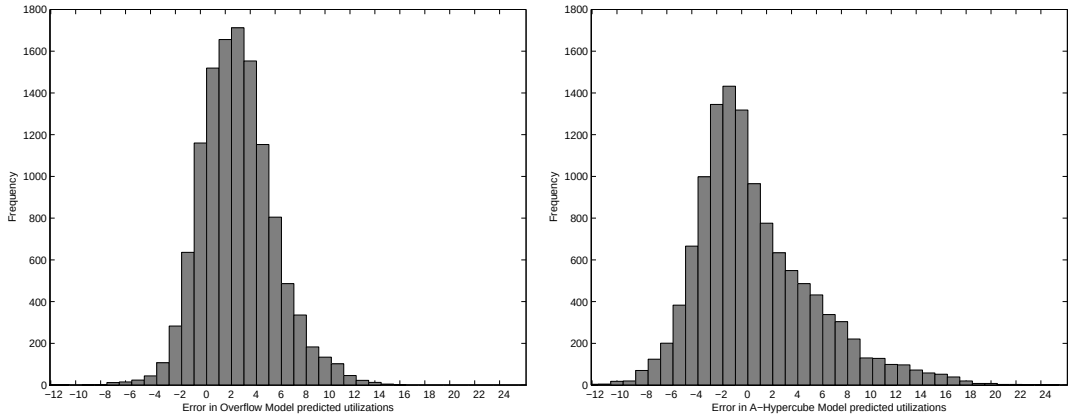

Figure 2.5: Differences between vehicle utilizations obtained from the overflow model and the simulation model (left). Corresponding differences for the A-Hypercube model.

in this implementation each iteration of the A-Hypercube is faster by a factor of 30. This is expected considering that each iteration of the overflow method updates a whole matrix of probabilities of size $J \times B$, whereas the A-Hypercube method deals with a vector of size N, which is generally much smaller. We also note that the MATLAB implementation of the A-Hypercube method was easy to vectorize while this was not possible for the overflow method. We anticipate that the speed ratio of C language implementations of the methods would be closer to 10:1 or even less. The average number of iterations taken by each of the methods to converge was, for $\epsilon = 10^{-6}$, 11.7±2.1 (A-Hypercube) and 16.9±1.7 (overflow). For $\epsilon = 10^{-4}$, we got means of 6.7 and 10.9 iterations, respectively. The A-Hypercube method failed to converge in 2 out of the 1000 allocations studied while the overflow method converged for all of them.

Finally, we note that both ways of initializing the matrix S^0 discussed in Appendix B led to the same fixed point in all cases. We conjecture that the function f implicitly defined by (A.1)-(A.6) has a unique fixed point, but this is

still an open question.

2.6 Conclusions and Future Research

We introduced two models for the static ambulance deployment problem. The models capture some of the essential queueing dynamics of an emergency medical service system while allowing efficient solution procedures. The island model is particularly suitable when analyzing multi-region systems managed by a central planner. It illustrates that ambulances should not be allocated in proportion to the loads offered to the locations, especially when ambulance utilizations are low. It is primarily intended to provide insight into ambulance-allocation decisions, and its recommendations should be viewed as a starting point for further exploration of allocations.

The overflow model estimates the loads offered to the different bases through a system of equations involving the Erlang loss formula. Whether this system of equations has a unique solution or not is an open question. Computational experiments suggest that the predictions of the second model are quite accurate, and the model is particularly useful for evaluating a large set of possible ambulance allocations and selecting some allocations to be further evaluated through simulation. The method is computationally more demanding than the A-Hypercube model, but still orders-of-magnitude faster than simulation. Its predictions are more accurate than those of the A-Hypercube for light to moderate ambulance utilization, while the situation is reversed for moderate to heavy ambulance utilization.

It would be interesting to investigate whether our models can incorporate

the fact that the average service time for an ambulance stationed at a base is affected by the location of the demand assigned to it. Also, provided the necessary input data is available, it would be straightforward to take random travel times into account in our analysis and test whether this feature improves the accuracy of the predictions. In particular, if the distribution of travel times from any base k to any node j is known, one can compute any steady state performance measure that is a function of the travel time to the call by conditioning on j and k in Equation (2.5).

An important assumption underlying both of our models is that demand is independent at different locations. This seems reasonable when the ambulance service is operating under typical conditions. But the assumption is violated when one considers the possibility of catastrophic events that strike multiple locations simultaneously. To effectively plan for such situations, it would be necessary to consider fundamentally different strategies.

In summary, we have developed analytical tools to aid EMS system managers in their quantitative evaluations of ambulance allocation decisions. We believe that our tools will be most valuable in pointing out the promising ambulance allocations. In this sense, we do not expect our tools to completely replace simulation, but rather to complement it, by enabling pre-screening and fast discarding of many poor ambulance allocations.

2.7 Acknowledgments

We thank Armann Ingolfsson for the data used for the numerical experiments in Section 2.5, and the referees for very helpful comments that improved both

the content and exposition in the paper.

This manuscript is based upon work supported by the National Science Foundation Grants No. DMI 0400287 and DMI 0422133. All opinions, findings, and conclusions or recommendations expressed in this manuscript are those of the authors and do not necessarily reflect the views of the National Science Foundation.

CHAPTER 3

APPROXIMATE DYNAMIC PROGRAMMING FOR AMBULANCE REDEPLOYMENT

> Normal people ... believe that if it ain't broke, don't fix it. Engineers believe that if it ain't broke, it doesn't have enough features yet.
>
> Scott Adams

We present an approximate dynamic programming (ADP) approach for making ambulance redeployment decisions in an emergency medical service system. The primary decision is where we should redeploy idle ambulances so as to maximize the number of calls reached within a given delay threshold. In section 3.1 we review past literature on the subject of ambulance redeployment and on the recent boom in successful applications of approximate dynamic techniques. We also comment on how our approach differentiates itself from previous works. In section 3.2 we give a formulation of dynamic redeployment as a dynamic program in a high-dimensional uncountable state space. To deal with such a space, we adopt an ADP approach (Section 3.3) whose main ingredient is a linear approximation architecture to the value function consisting of 6 basis functions that is parameterized by a small number of parameters. The details on the architecture are presented in Section 3.4 We tune the parameters of the value function approximations using simulated cost trajectories of the system. Computational experiments, presented in Sections 3.5 and 3.6 demonstrate the performance of the approach on emergency medical service systems in two metropolitan areas. We report practically significant improvements in

performance relative to benchmark static policies. We conclude in Section 3.7 and point out some directions for further research.

3.1 Literature review

There are two streams of literature that are related to our work. The first one is the literature on ADP. A generic approach for ADP involves using value function approximations of the form $\sum_{p=1}^{P} r_p \phi_p(\cdot)$, where $\{r_p : p = 1, \ldots, P\}$ are tuneable parameters and $\{\phi_p(\cdot) : p = 1, \ldots, P\}$ are fixed basis functions; see Bertsekas and Tsitsiklis [12], Powell [48]. There are a number of methods to tune the parameters $\{r_p : p = 1, \ldots, P\}$ so that $\sum_{p=1}^{P} r_p \phi_p(\cdot)$ yields a good approximation to the value function. For example, temporal difference learning and Q-learning use stochastic approximation ideas in conjunction with simulated trajectories of the system to iteratively tune the parameters; see Sutton [55], Watkins and Dayan [63], Tsitsiklis [59], Bertsekas and Tsitsiklis [12], Tsitsiklis and Van Roy [60], Si, Barto, Powell and Wunsch II [53]. On the other hand, the linear-programming approach for ADP finds a good set of values for the parameters by solving a large linear program whose decision variables are $\{r_p : p = 1, \ldots, P\}$; see Schweitzer and Seidmann [52], de Farias and Van Roy [18], Adelman and Mersereau [3]. For more on the linear-programming approach, see Chapter 4. Both classes of approaches are aimed at tuning the parameters $\{r_p : p = 1, \ldots, P\}$ so that $\sum_{p=1}^{P} r_p \phi_p(\cdot)$ yields a good approximation to the value function. The choice of the basis functions $\{\phi_p(\cdot) : p = 1, \ldots, P\}$, on the other hand, is regarded as more of an art form, requiring substantial knowledge of the problem structure. Applications of ADP include inventory control [62], inventory routing [1], option pricing [61], game playing [65, 22], dynamic fleet

management [57], and network revenue management [2, 23].

The second stream of literature that is related to our work is the literature on ambulance redeployment. One class of models involves solving integer programs in real time whenever an ambulance redeployment decision needs to be made; see Kolesar and Walker [41], Gendreau, Laporte and Semet [25], Brotcorne et al. [14], Gendreau, Laporte and Semet [26], Nair and Miller-Hooks [47], Richards [51]. The objective function in these integer programs generally involves a combination of backup coverage for future calls and relocation cost of ambulances. These models are usually computationally intensive, since they require solving an optimization problem every time a decision is made. As a result, a parallel computing environment is often (but not always) used to implement a working real-time system. A second class of models is based on solving integer programs in a preparatory phase. This approach provides a lookup table describing, for each number of available ambulances, where those ambulances should be deployed. Dispatchers attempt to dispatch so as to keep the ambulance configuration close to the one suggested by the lookup table; see Ingolfsson [35], Goldberg [28]. A third class of models attempts to capture the randomness in the system explicitly, either through a dynamic programming formulation or through heuristic approaches. Berman [7], Berman [8] and Berman [9] represent the first papers that provide a dynamic programming approach for the ambulance redeployment problem. However, these papers follow an exact dynamic programming formulation and, as is often the case, this formulation is tractable only in oversimplified versions of the problem with few vehicles and small transportation networks. Andersson [4] and Andersson and Vaerband [5] make the ambulance redeployment decision by using a "preparedness" function that essentially measures the capability of a certain ambulance config-

uration to cover future calls. The preparedness function is similar in spirit to the value function in a dynamic program, measuring the impact of current decisions on the future evolution of the system. However, the way the preparedness function is constructed is heuristic in nature.

When compared with the three classes of models described above, our approach provides a number of advantages. In contrast to the models that are based on integer programs, our approach captures the random evolution of the system over time since it is based on a stochastic dynamic programming formulation of the ambulance redeployment problem. Furthermore, the decisions made by our approach in real time can be computed very quickly as this requires solving a simple optimization problem that minimizes the sum of the immediate cost and the value function approximation. In lookup table approaches, there may be more than one way to redeploy the ambulances so that the ambulance configuration over the transportation network matches the configuration suggested by the lookup table. Therefore, table lookup approaches still leave some aspects of dispatch decisions to subjective interpretation by dispatchers. Our approach, on the other hand, can fully automate the decision-making process. In traditional dynamic-programming approaches, one is usually limited to very small problem instances, whereas ADP can be used on problem instances with realistic dimensions. Our approach allows working with a variety of objective functions, such as the number of calls that are not served within a threshold time standard or the total response time for the calls. Furthermore, our approach allows the possibility of constraining the frequency and destinations of ambulance relocations. This is important since a relocation scheme should balance improvements in service levels with the additional demands imposed on ambulance crews.

3.2 Ambulance Redeployment as a Markov Decision Process

In this section, we present a dynamic programming formulation of the ambulance redeployment problem. As will shortly be clear, our model involves an uncountable state space. For an excellent account of the basic terminology, notation and fundamental results regarding dynamic programming in uncountable states spaces, we refer the reader to [11].

3.2.1 State Space

Two main components in the state of an EMS system at a given time are the vectors $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$ and $\mathbf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$ containing information about the state of each the N ambulances and each of the M queued calls in the system. To simplify the presentation, we assume that we do not keep more than M waiting calls in the system, possibly by diverting them to another EMS system. This is not really a restriction from a practical perspective since M can be quite large. The state of each ambulance is given by a tuple $\mathbf{a} = (\sigma_a, o_a, d_a, t_a)$, where σ_a is the status of the ambulance, o_a and d_a are respectively origin and destination locations of the ambulance and t_a is the starting time of the ambulance movement. To serve a call, an ambulance first moves to the call scene and provides service at the scene for a certain amount of time. Following this, the ambulance possibly transports the patient to a hospital, and after spending some time at the hospital, the ambulance becomes free to serve another call. Therefore, the status of an ambulance σ_a can be "off shift," "idle at base," "going to call," "at call scene," "going to hospital," "at hospital" or "returning to base." If the ambulance is stationary at location o, then we have $o_a = d_a = o$. If the ambulance is in

motion, then t_a corresponds to the starting time of this movement. Otherwise, t_a corresponds to the starting time of the current phase in the service cycle. For example, if the status of the ambulance is "at hospital," then t_a corresponds to the time at which the ambulance arrived at the hospital. This time is kept in the state of an ambulance to be able to give a Markov formulation for the non-Markovian elements in the system, such as nonexponentially distributed service times and deterministic travel times. Similarly, for a call, we have $\mathbf{c} = (\sigma_c, o_c, t_c, p_c)$, where σ_c is the status of the call, o_c is the location of the call, t_c is the time at which the call arrived into the system and p_c is the priority level of the call. The status of a call σ_c can be "assigned to an ambulance" or "queued for service."

We model the dynamics of the system as an event-driven process. Events are triggered by changes in the status of the ambulances or by call arrivals. Therefore, the possible event types in the system are "call arrives," "ambulance goes off shift," "ambulance comes on shift," "ambulance departs for call scene," "ambulance arrives at call scene," "ambulance leaves call scene," "ambulance arrives at hospital," "ambulance leaves hospital" and "ambulance arrives at base." We assume that we can make decisions (for any ambulance, and not just the one involved in the event) only at the times of these events. This comes at the cost of some loss of optimality, since making decisions between the times of the events may improve performance. From a practical perspective, however, events are frequent enough to allow ample decision-making opportunities.

By restricting our attention to the times of events, we visualize the system as jumping from one event time to another. Therefore, we can use the tuple $s = (\tau, e, \mathbf{A}, \mathbf{C})$ to represent the state of the system, where τ corresponds to the current time, e corresponds to the current event type, and $\mathbf{A}$ and $\mathbf{C}$ respectively

correspond to the state of the ambulances and the calls. In this case, the state trajectory of the system can be written as $\{s_k : k = 1, 2, \ldots\}$, where s_k is the state of the system just after the kth event occurs. Note that the time is rolled into our state variable. Throughout the paper, we use $\tau(s)$ and $e(s)$ to respectively denote the time and the event type when the state of the system is s. In other words, $\tau(s)$ and $e(s)$ are the first two components of the tuple $s = (\tau, e, \mathbf{A}, \mathbf{C})$.

3.2.2 Controls

We assume that calls are served in decreasing order of priority, and within a given priority level are served in first-in-first-out order. We further assume that the closest available ambulance is dispatched to a call. This is not an exact representation of reality, but is close enough for our purposes: We will show that ADP yields an effective redeployment strategy under this dispatch policy.

Let $\mathcal{R}(s)$ denote the set of ambulances that are available for redeployment when the state of the system is s. Let $u_{ab}(s) = 1$ if we redeploy ambulance a to base b when the state of the system is s, and 0 otherwise. Letting $\mathcal{B}$ be the set of ambulance bases, we can capture the potential reallocation decisions in the binary matrices $u(s) = \{u_{ab}(s) : a \in \mathcal{R}(s),\ b \in \mathcal{B}\}$. If we allow a maximum of $m(s)$ ambulance relocations at this state, then set of feasible decisions is then

$$\mathcal{U}(s) = \Big\{u \in \{0,1\}^{|\mathcal{R}(s)|\times|\mathcal{B}|} : \sum_{a\in\mathcal{A}, b\in Bcal} u_{ab}u_{ab} \le m(s), \quad \sum_{b\in\mathcal{B}} u_{ab} \le 1 \quad \forall\, a \in \mathcal{R}(s)\Big\}.$$

The constraints in this definition simply state that each ambulance that is considered for redeployment can be redeployed to at most one base. An important assumption is that the cardinality of $\mathcal{U}(s)$ is small so that an optimization problem over this feasible set can be solved by enumerating over all feasible

solutions. This assumption is met if $m(s)$ is kept small, say 1 or 2.

We can use different definitions of $\mathcal{R}(s)$ to permit different amounts of relocation. For example, all available ambulances are considered for relocation if $\mathcal{R}(s)$ denotes the set of ambulances that are either idle at base or returning to base. But this may lead to impractically many relocations, so we can restrict $\mathcal{R}(s)$, for example to $\varnothing$ unless the event $e(s)$ corresponds to an ambulance becoming free after serving a call, in which case we can take $\mathcal{R}(s)$ equal to the singleton set corresponding to the newly freed ambulance. Here the ambulance crews are "on the road" when considered for relocation so there is no additional disruption to the crews relative to the standard "return to base" policy.

3.2.3 Fundamental Dynamics

Call arrivals are generated across the region $R \subset \mathbb{R}^2$ according to a Poisson point process with a known arrival intensity $C = \{C(t, x, y) : t \geq 0,\ (x, y) \in R\}$. As mentioned above, we have a fixed policy for serving calls, but our general approach does not depend on the particular form of this policy. If there are no available ambulances to serve a call, then the call is placed into a waiting queue. An ambulance serving a call proceeds through a sequence of events, including arriving at the scene, treating the patient, transporting and handing over the patient at the hospital. The main source of uncertainty in this call service cycle are the times spent between events. We assume that probability distributions for all of the event durations are known.

Besides these sources of randomness, the major ingredient governing the dynamics is the decision-making of dispatchers. As a result, the complete tra-

jectory of the system is given by $\{(s_k, u_k) : k = 1, 2, \ldots\}$, where s_k is the state of the system at the time of the kth event and u_k is the decision (if any) made by the dispatcher when the state of the system is s_k. We capture the dynamics of the system symbolically by

$$s_{k+1} = f(s_k, u_k, X_k(s_k, u_k)),$$

where $X_k(s_k, u_k)$ is a random element of an appropriate space encapsulating all the sources of randomness described above. We do not attempt to give a more explicit description of the transition function $f(\cdot, \cdot, \cdot)$, since this does not add anything of value to our treatment. It suffices to point out that the preceding discussion and the fact that we can simulate the evolution of the system over time imply that an appropriate $f(\cdot, \cdot, \cdot)$ exists.

3.2.4 Transition Costs

Along with a transition from state s_k to s_{k+1} through decision u_k, we incur a cost $g(s_k, u_k, s_{k+1})$. In our particular implementation, we use the cost function

$$g(s_k, u_k, s_{k+1}) = \begin{cases} 1 & \text{if the event } e(s_{k+1}) \text{ corresponds to an ambulance arrival} \\ & \text{at a call scene, the call in question is urgent and} \\ & \text{the response time exceeds } \Delta \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Here Δ is a fixed threshold response time that is on the order of 10 minutes. Therefore, the cost function (3.1) allows us to count the number of high priority calls whose response times exceed a threshold response time. We are interested

in the performance of the system over the finite planning horizon $[0, T]$, so let $g(s, \cdot, \cdot) = 0$ whenever $\tau(s) > T$. In our implementation, T corresponds to two weeks. We have chosen this particular cost function because of its simplicity, and also because most performance benchmarks in the EMS industry are formulated in terms of the percentage of calls that are reached within a time standard. Our approach allows other cost functions without affecting the general treatment.

3.2.5 Objective Function and Optimality Equation

A policy is a mapping from the state space to the action space, prescribing which action to take for each possible state of the system. Throughout the paper, we use $\mu(s)$ to denote the action prescribed by policy μ when the state of the system is s. If we follow policy μ, then the state trajectory of the system $\{s_k^\mu : k = 1, 2, \ldots\}$ evolves according to $s_{k+1}^\mu = f(s_k^\mu, \mu(s_k^\mu), X_k(s_k^\mu, \mu(s_k^\mu)))$ and the discounted total expected cost incurred by starting from initial state s is given by

$$J^\mu(s) = \mathbb{E}\Big[\sum_{k=1}^{\infty} \alpha^{\tau(s_k^\mu)}\, g(s_k^\mu, \mu(s_k^\mu), s_{k+1}^\mu) \,|\, s_1^\mu = s \Big],$$

where $\alpha \in [0, 1)$ is a fixed discount factor. Below we discuss the reasons for using discounting. The expectation in the expression above involves the random variables $\{s_k^\mu : k = 1, 2, \ldots\}$ and $\tau(s_k^\mu)$ is the time at which the system visits the kth state. It is well-known that the policy μ^* that minimizes the discounted total expected cost can be found by computing the value function through the optimality equation

$$J(s) = \min_{u \in \mathcal{U}(s)} \left\{ \mathbb{E}\Big[g(s, u, f(s, u, X(s, u))) + \alpha^{\tau(f(s,u,X(s,u))) - \tau(s)}\, J(f(s, u, X(s, u))) \Big] \right\} \quad (3.2)$$

and letting $\mu^*(s)$ be the minimizer of the right-hand side above; see Bertsekas and Shreve [11].

The difficulty with the optimality equation (3.2) is that the state variable takes uncountably many values. Even if we are willing to discretize the state variable to obtain a countable state space, the state variable is still a high-dimensional vector and solving the optimality equation (3.2) through classical dynamic-programming approaches is computationally very difficult. In the next two sections, we propose a method to construct tractable approximations to the value function.

The discounting in (3.2) may seem odd, as it implies that we are minimizing the expected *discounted* number of calls that are not reached within the threshold time. This is purely a computational device, in the sense that the discount factor is very helpful in stabilizing the ADP approach that we describe in the next two sections. The key issue is that the effect of relocation decisions can be small relative to the simulation noise, and discounting appears to mitigate this to some extent. This observation is in agreement with empirical observations from the case studies in Chapter 8 of [12]. The increase in stability is also supported by the theoretical results in Chapter 6.2 of Bertsekas and Tsitsiklis [12]. When presenting our computational results in Sections 3.5 and 3.6, we report the *undiscounted* number of calls that are not reached within the threshold response time. These results indicate that although we construct the value function approximations with a view towards minimizing the expected *discounted* cost, the same value function approximations provide very good performance in minimizing the expected *undiscounted* cost.

3.3 Approximate Dynamic Programming

The ADP approach that we use to construct approximations to the value function is closely related to the traditional policy iteration algorithm in the Markov decision processes literature. We begin with a brief description of the policy iteration algorithm. Throughout the rest of the paper, the greedy policy induced by an arbitrary function $\hat{J}(\cdot)$ refers to the policy that takes a decision in the set

$$\underset{u \in \mathcal{U}(s)}{\operatorname{argmin}} \left\{ \mathbb{E}\Big[g(s, u, f(s, u, X(s, u))) + \alpha^{\tau(f(s,u,X(s,u)))-\tau(s)} \hat{J}(f(s, u, X(s, u))) \Big] \right\} \tag{3.3}$$

whenever the state of the system is s.

Policy Iteration

Step 1. Initialize the iteration counter n to 1 and initialize $J^1(\cdot)$ arbitrarily.

Step 2. (Policy improvement) Let μ^n be the greedy policy induced by $J^n(\cdot)$.

Step 3. (Policy evaluation) Let $J^{n+1}(\cdot) = J^{\mu^n}(\cdot)$, where $J^{\mu^n}(s)$ denotes the expected discounted cost incurred when starting from state s and using policy μ^n.

Step 4. Increase n by 1 and go to Step 2.

In our setting the cost function $g(\cdot, \cdot, \cdot)$ is nonnegative and the set of feasible decisions $\mathcal{U}(s)$ is finite, so Proposition 9.17 in Bertsekas and Shreve [11] applies and hence $J^n(\cdot)$ converges pointwise to the solution to the optimality equation (3.2).

The difficulty with the policy iteration algorithm is that when dealing with uncountable state spaces, it is not even feasible to store $J^{\mu^n}(\cdot)$, let alone compute the expected discounted cost incurred by using policy μ^n. We overcome this

difficulty by using value function approximations of the form

$$J(s,r) = \sum_{p=1}^{P} r_p \, \phi_p(s).$$

In the expression above, $r = \{r_p : p = 1, \ldots, P\}$ are tuneable parameters and $\{\phi_p(\cdot) : p = 1, \ldots, P\}$ are fixed basis functions. The challenge is to construct the basis functions and tune the parameters so that $J(\cdot, r)$ is a good approximation to the solution to the optimality equation (3.2). To achieve this, each basis function $\phi_p(\cdot)$ should capture some essential information about the solution to the optimality equation. In Section 3.4, we describe one set of basis functions that work well. Once a good set of basis functions is available, we can use the following approximate version of the policy iteration algorithm to tune the parameters $\{r_p : p = 1, \ldots, P\}$.

Approximate Policy Iteration

Step 1. Initialize the iteration counter n to 1 and initialize $r^1 = \{r_p^1 : p = 1, \ldots, P\}$ arbitrarily.

Step 2. (Policy improvement) Let μ^n be the greedy policy induced by $J(\cdot, r^n)$.

Step 3. (Policy evaluation through simulation) Simulate the trajectory of policy μ^n over the planning horizon $[0, T]$ for Q sample paths. Let $K(q)$ denote the number of states visited, $\{s_k^n(q) : k = 1, \ldots, K(q)\}$ the state trajectory of policy μ^n in sample path q and $G_k^n(q)$ be the discounted cost incurred by starting from state $s_k^n(q)$ and following policy μ^n in sample path q.

Step 4. (Projection) Let

$$r^{n+1} = \underset{r \in \mathbb{R}^P}{\operatorname{argmin}} \left\{ \sum_{q=1}^{Q} \sum_{k=1}^{K(q)} \left[G_k^n(q) - J(s_k^n(q), r) \right]^2 \right\}.$$

Step 5. Increase n by 1 and go to Step 2.

In Step 3 of approximate policy iteration we use simulation to evaluate the expected discounted cost incurred by policy μ^n. Therefore, $\{G_k^n(q) : k = 1, \ldots, K(q),\ q = 1, \ldots, Q\}$ are the sampled cost trajectories of the system under policy μ^n. In Step 4, we tune the parameters so that the value function approximation $J(\cdot, r)$ provides a good fit to the sampled cost trajectories.

There is still one computational difficulty in the approximate policy iteration algorithm. When simulating the trajectory of policy μ^n in Step 3, we need to solve an optimization problem of the form (3.3) to find the action taken by the greedy policy induced by $J(\cdot, r^n)$. This optimization problem involves an expectation that is difficult to compute. We resort to Monte Carlo simulation to overcome this difficulty. In particular, if the state of the system is s and we want to find the action taken by the greedy policy induced by $J(\cdot, r^n)$ in this state, then we enumerate over all decisions in the feasible set $\mathcal{U}(s)$. Starting from state s and taking decision u, we simulate the trajectory of the system until the next event and this provides a sample of $f(s, u, X(s, u))$. By using multiple samples, we estimate the expectation

$$\mathbb{E}\Big[g(s, u, f(s, u, X(s, u))) + \alpha^{\tau(f(s,u,X(s,u)))-\tau(s)} J(f(s, u, X(s, u)), r^n)\Big]$$

through a sample average. Once we estimate the expectation above for all $u \in \mathcal{U}(s)$, we choose the decision that yields the smallest value and use it as the decision taken by the greedy policy induced by $J(\cdot, r^n)$ when the state of the system is s. This approach is naturally subject to sampling error, but it provides good performance in practice.

Proposition 6.2 in [12] provides a performance guarantee for the approximate policy iteration algorithm. (The result is easily extended from finite to infinite state spaces.) This is an encouraging result as it provides theoretical

support for the approximate policy iteration algorithm, but its conditions are difficult to verify in practice. In particular, the result assumes that we precisely know the error induced by using regression to estimate the discounted total expected cost of a policy, and it assumes that expectations are computed exactly rather than via sampling as in our case. For this reason, we do not go into the details of this result and refer the reader to Bertsekas and Tsitsiklis [12] for further details.

3.4 Basis Functions

In this section, we describe the basis functions $\{\phi_p(\cdot) : p = 1, \ldots, P\}$ that we use in our value function approximations. We use $P = 6$ basis functions, some of which are based on the queueing insights developed in Restrepo, Henderson and Topaloglu [49].

3.4.1 Baseline

The first basis function is of the form $\phi_1(s) = 1 - \alpha^{T-\tau(s)}$. The role of this basis function is to give a very rough estimate of the discounted number of missed calls between the time associated with state s and the end of the planning horizon. In particular, if we assume that we miss a call every θ time units, then the discounted number of missed calls over the interval $[\tau(s), T]$ is

$$1 + \alpha^{\theta} + \alpha^{2\theta} + \ldots + \alpha^{\lfloor \frac{T-\tau(s)}{\theta} \rfloor \theta} = \frac{1 - \alpha^{\left\lceil \lfloor \frac{T-\tau(s)}{\theta} \rfloor + 1 \right\rceil \theta}}{1 - \alpha} \approx \frac{1 - \alpha^{T-\tau(s)}}{1 - \alpha}.$$

3.4.2 Unreachable Calls

The second basis function computes the number of waiting calls for which an ambulance assignment has been made, but the ambulance will not reach the call within the threshold response time. This is easily computed in our setting where travel times are deterministic. In the case of stochastic travel times, we could use the probability that the ambulance will reach the call within the threshold response time instead. We do not give a precise expression for this basis function, since the precise expression is cumbersome yet straightforward to define.

3.4.3 Uncovered Call Rate

The third basis function captures the rate of call arrivals that cannot be reached on time by any available ambulance. To define this precisely we need some additional notation. Recall that calls arrive across the region $R \subset \mathbb{R}^2$ according to a Poisson point process with arrival intensity $C = \{C(t, x, y) : t \geq 0,\ (x, y) \in R\}$. Partition the region R into L subregions $\{\rho_l : l = 1, \ldots, L\}$, and associate a representative point or "center of mass" (x_l, y_l) with each subregion $l = 1, \ldots, L$.

The coverage of subregion l is the number of available ambulances that can reach the center of mass (x_l, y_l) within the threshold time standard. Let $d(t, (x_1, y_1), (x_2, y_2))$ be the travel time between points (x_1, y_1) and (x_2, y_2) when travel starts at time t. Then, letting $\mathbf{1}(\cdot)$ be the indicator function, the coverage of subregion l can be written as a function of the state of the system as

$$N_l(s) = \sum_{a \in \mathcal{A}(s)} \mathbf{1}(d(\tau(s), (x_a(s), y_a(s)), (x_l, y_l)) \leq \Delta).$$

Here, $\mathcal{A}(s)$ is the set of available ambulances and $(x_a(s), y_a(s))$ is the location of ambulance a at time $\tau(s)$ when the system is in state s. The rate of call arrivals at time t in subregion l is

$$C_l(t) = \int_{\rho_l} C(t, x, y)\, dx\, dy.$$

Therefore, when the state of the system is s, we can compute the rate of call arrivals that are not covered by any available ambulance by

$$\phi_3(s) = \sum_{l=1}^{L} C_l(\tau(s))\, \mathbf{1}(N_l(s) = 0).$$

3.4.4 Missed Call Rate

The previous 2 basis functions capture calls already received that we know we cannot reach on time, and the rate of arriving calls that cannot be reached on time because they are too far from any available ambulance. We could also fail to reach a call on time due to queueing effects from ambulances being busy with other calls. The fourth basis function represents an attempt to capture this effect. This basis function is of the form

$$\phi_4(s) = \sum_{l=1}^{L} C_l(\tau(s))\, P_l(s),$$

where $P_l(s)$ is the probability that all ambulances that could reach a call in subregion l on time are busy with other calls. We estimate $\{P_l(s) : l = 1, \ldots, L\}$ by treating the call service processes in different subregions as Erlang-loss systems. In Erlang-loss systems, calls arriving when all servers are busy are lost. In reality such calls are queued and served as ambulances become free, but the time threshold is almost always missed for such calls, so counting them as lost seems reasonable. The issue that such calls impose some load on the true system but are discarded in an Erlang-loss system creates a slight mismatch, but

our computational results show that this function is still highly effective as a basis function.

The Erlang-loss probability $\mathcal{L}(\lambda, \mu, c)$ for a system with arrival rate λ, service rate μ and c servers is

$$\mathcal{L}(\lambda, \mu, c) = \frac{(\lambda/\mu)^c/c!}{\sum_{i=0}^{c}(\lambda/\mu)^i/i!}.$$

To characterize the Erlang-loss system for subregion l, given that the state of the system is s, we need to specify the number of servers, and the arrival and service rates. Let $\mathcal{N}_l(s)$ be the set of available ambulances that can serve a call in subregion l within the threshold response time. Then

$$\mathcal{N}_l(s) = \{a \in \mathcal{A}(s) : d(\tau(s), (x_a(s), y_a(s)), (x_l, y_l)) \leq \Delta\}.$$

We use $c = |\mathcal{N}_l(s)|$ as the number of servers in the Erlang loss system for subregion l. Let $\mu_l(t)$ be the service rate in the loss system, i.e., the rate at which an ambulance can serve a call at time t in subregion l. It is difficult to come up with a precise value for $\mu_l(t)$. It primarily depends on the time spent at the scene of a call and any transfer time at a hospital, since travel times are usually small relative to these quantities. In our practical implementation, we use historical data to estimate the time spent at the call scenes and the hospitals and add a small padding factor to capture travel times. Finally, let $\Lambda_l(s)$ be the rate of call arrivals that should be served by ambulances in the set $\mathcal{N}_l(s)$. Coming up with a value for $\Lambda_l(s)$ is even more difficult than devising a value for $\mu_l(t)$. One option is to let $\Lambda_l(s) = C_l(\tau(s))$, which is the rate of call arrivals at time $\tau(s)$ in subregion l. However, ambulances in the set $\mathcal{N}_l(s)$ serve not only calls in subregion l, but also calls in other subregions. To attempt to capture this let

$$\Lambda_l(s) = \sum_{a \in \mathcal{N}_l(s)} \sum_{i=1}^{L} C_i(\tau(s))\, \mathbf{1}(d(\tau(s), (x_a(s), y_a(s)), (x_i, y_i)) \leq \Delta), \tag{3.4}$$

so that $\Lambda_l(s)$ reflects the total call arrival rate in subregions that are close to any of the ambulances in the set $\mathcal{N}_l(s)$. We then use the approximation

$$P_l(s) \approx \mathcal{L}(\Lambda_l(s), \mu_l(\tau(s)), |\mathcal{N}_l(s)|). \tag{3.5}$$

There are at least 2 shortcomings in the estimate for $\Lambda_l(s)$ in (3.4) and the approximation to $P_l(s)$ in (3.5). First, there is double counting in the estimate of $\Lambda_l(s)$. In particular, if 2 ambulances $a_1, a_2 \in \mathcal{N}_l(s)$ can both reach subregion ℓ within the time threshold, then the summation for $\Lambda_l(s)$ counts $C_\ell(\tau(s))$ twice. Second, $C_\ell(\tau(s))$ could be counted in the demand rates for multiple subregions. To be more precise, if there are three subregions l_1, l_2, ℓ and two ambulances $a_1 \in \mathcal{N}_{l_1}(s)$, $a_2 \in \mathcal{N}_{l_2}(s)$ such that both a_1 and a_2 can reach subregion ℓ within the time threshold, then the summations for $\Lambda_{l_1}(s)$ and $\Lambda_{l_2}(s)$ both count $C_\ell(\tau(s))$. Therefore, we typically have $\sum_{l=1}^{L} \Lambda_l(s) > \sum_{l=1}^{L} C_l(\tau(s))$. To overcome this problem, we dilute the call arrival rates by a factor κ. In particular, we use the call arrival rate $\kappa C_\ell(\tau(s))$ in (3.4) instead of $C_\ell(\tau(s))$ so that we may roughly have $\sum_{l=1}^{L} \Lambda_l(s) = \sum_{l=1}^{L} \kappa C_l(\tau(s))$. We find a good value for κ through preliminary experimentation. Interestingly, as we will see in our computational results, it is not necessarily the case that the best choice of κ lies in $(0, 1)$. We emphasize that κ is the only tuneable parameter in our ADP approach that requires experimentation.

3.4.5 Future Uncovered Call Rate

In certain settings, we may not be willing to redeploy ambulances that are already in transit to a particular location. In this case, from the perspective of covering future calls, the ambulance destinations are as important as their current

locations. This is the motivation underlying the fifth and sixth basis functions. Our fifth basis function parallels the third one, replacing the true locations of ambulances by their destinations.

The definition of this basis function is therefore identical to that of ϕ_3, but the configuration of the ambulances that we use to compute $N_l(s)$ is not the current one, but rather, an estimated future configuration that is obtained by letting all ambulances in transit reach their destinations and all stationary ambulances remain at their current locations. To be precise, given that the system is in state $s = (\tau, e, \mathbf{A}, \mathbf{C})$ with $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$ and $\mathbf{a}_i = (\sigma_{a_i}, o_{a_i}, d_{a_i}, t_{a_i})$, we define a new state $\vec{s}(s) = (\tau + 1/\sum_{l=1}^{L} C_l(\tau), e, \vec{\mathbf{A}}, \mathbf{C})$ with $\vec{\mathbf{A}} = (\vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_N)$ and $\vec{a}_i = (\vec{\sigma}_{a_i}, d_{a_i}, d_{a_i}, \tau + 1/\sum_{l=1}^{L} C_l(\tau))$, where $\vec{\sigma}_{a_i}$ is the status of ambulance a_i when it reaches its destination d_{a_i}. In this case, we can write the fifth basis function as

$$\phi_5(s) = \sum_{l=1}^{L} C_l(\tau(\vec{s}(s)))\, \mathbf{1}(N_l(\vec{s}(s)) = 0).$$

The time $\tau(\vec{s}(s)) = \tau + 1/\sum_{l=1}^{L} C_l(\tau)$ is used as an approximation to the expected time of the next call arrival. The next call may arrive before or after the ambulances actually reach their destinations, but we heuristically use the time $\tau(\vec{s}(s))$ simply to look into the future. The idea is that the estimated future configuration of the ambulances $\vec{\mathbf{A}}$ is more likely to hold at the future time $\tau(\vec{s}(s))$ than at the current time $\tau(s)$.

3.4.6 Future Missed Call Rate

The sixth basis function, ϕ_6, parallels ϕ_4 in that it captures the rate of calls that will likely be lost due to queueing congestion. As with the fifth basis function,

it uses an estimated future configuration of the ambulances. It is defined as

$$\phi_6(s) = \sum_{l=1}^{L} C_l(\tau(\vec{s}(s)))\, P_l(\vec{s}(s)).$$

3.5 Computational Results on Edmonton

In this section, we present computational results for the EMS system in the city of Edmonton, Alberta in Canada. We begin with a description of the data set along with our assumptions.

3.5.1 Experimental Setup

Our data are based on the data set used in the computational experiments in [36]. The city of Edmonton has a population of 730,000 and covers an area of around 40×30 km^2. The EMS system includes 16 ambulances, 11 bases and 5 hospitals. We assume for simplicity that all ambulances are of the same type and operate all day. An ambulance, upon arriving at a call scene, treats the patient for an exponentially distributed amount of time with mean 12 minutes. After treating the patient at the call scene, the ambulance transports the patient to a hospital with probability 0.75. The probability distribution for the hospital chosen is inferred from historical data. The time an ambulance spends at the hospital has a Weibull distribution with mean 30 minutes and standard deviation 13 minutes. The turn-out time for the ambulance crews is 45 seconds. In other words, if the ambulance crew is located at a base when it is notified of a call, then it takes 45 seconds to get ready. An ambulance crew already on the road does not incur the turn-out time. The road network that we use in

our computational experiments models the actual road network on the avenue level. There are 252 nodes and 934 arcs in this road network. The travel times are deterministic and do not depend on the time of the day.

There was not enough historical data to develop a detailed model of the call arrivals so we proceeded with an insightful, but not necessarily realistic, model. The model maintains a constant overall arrival rate, but the distribution of the location of calls changes in time. We divided the city of Edmonton into 20×17 subregions and assumed that the rate of call arrivals in subregion l at time t is given by

$$C_l(t) = C[\gamma_l + \delta_l \sin(2\pi t/24)],$$

where t is measured in hours. In the expression above, C, γ_l and δ_l are fixed parameters that satisfy $C \geq 0$, $\gamma_l \geq |\delta_l|$, $\sum_{l=1}^{340} \gamma_l = 1$ and $\sum_{l=1}^{340} \delta_l = 0$. We have $\sum_{l=1}^{340} \gamma_l + \sum_{l=1}^{340} \delta_l \sin(2\pi t/24) = 1$ so that we can interpret C as the total call arrival rate into the system and $\gamma_l + \delta_l \sin(2\pi t/24)$ as the probability that a call arriving at time t falls in subregion l. If $\delta_l > 0$, then the peak call arrival rate in subregion l occurs at hours $\{6, 30, 54, \ldots\}$, whereas if $\delta_l < 0$, then the peak call arrival rate in subregion l occurs at hours $\{18, 42, 66, \ldots\}$. The average call arrival rate over a day in subregion l is $C\gamma_l$. We estimated C and γ_l using historical data and C ranged from 3 to 7 calls per hour. We chose appropriate values of δ_l so that we have higher call arrival rates in the business subregions early in the day and higher call arrival rates in the residential subregions later in the day.

We implemented ADP on top of the BartSim simulation engine developed by [34]. In all of our computational experiments, we use a discount factor of $\alpha = 0.85$ per day. The simulation horizon is 14 days. We use 30 sample paths in Step 3 of the approximate policy iteration algorithm. After some experimentation,

we found that $\kappa = 0.1$ in the fourth and sixth basis functions gave the best results.

We use the static redeployment policy as a benchmark. In particular, the static redeployment policy preassigns a base to each ambulance and redeploys each ambulance back to its preassigned base whenever it becomes free after serving a call. We found a good static redeployment policy by simulating the performance of the system under a large number of possible base assignments and choosing the base assignment that gave the best performance.

3.5.2 Baseline Performance

In our first set of computational experiments, we test the performance of ADP under the assumption that the only time we can redeploy an ambulance is when it becomes free after serving a call. We do not redeploy ambulances at other times. Following the discussion of Section 3.2.2, this is equivalent to setting $\mathcal{R}(s) = \varnothing$, except when $e(s)$ corresponds to an ambulance becoming free after serving a call, in which case $\mathcal{R}(s)$ is the singleton set corresponding to the ambulance just becoming free. The motivation for using this redeployment strategy is that it minimizes the disturbance to the crews and never redeploys an ambulance that is located at an ambulance base.

Figure 3.5.2 shows the performance of ADP. The horizontal axis gives the iteration number in the approximate policy iteration algorithm and the vertical axis gives the percentage of calls not reached within the threshold response time. The plot gives the percentage of calls missed by the greedy policy induced by the value function approximation at a particular iteration. The dashed horizontal line shows the best performance attained by ADP, while the thin hori-

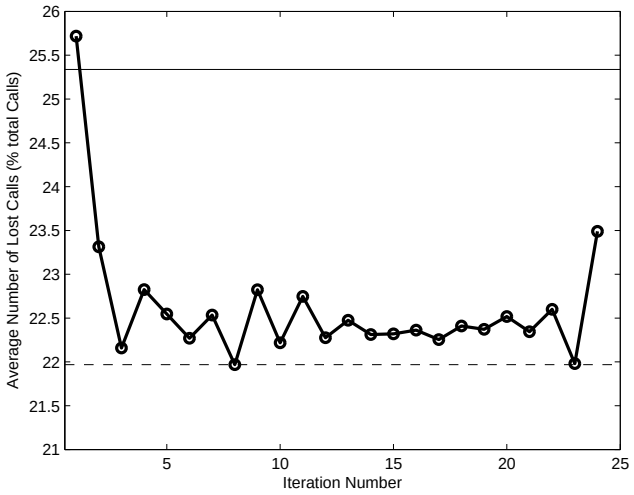

Figure 3.1: Performance of ADP and the benchmark policy (solid thin line).

zontal line shows the performance of the best static policy. ADP attains a good policy within two iterations and then bounces around this policy. The best ADP policy misses 21.9% (± 0.2%) of the calls, whereas the best static redeployment policy misses 25.3% (±0.2%) of the calls. (These figures are undiscounted numbers of missed calls.)

The ADP approach does not converge on a single policy. This lack of convergence is not a concern from a practical viewpoint since we can simply choose the best policy that is obtained during the course of the approximate policy iteration algorithm and implement this policy in practice.

To emphasize the importance of choosing the basis functions carefully, Figure 3.5.2 shows the performance of ADP when we use an arrival rate $\lambda = C_l(\tau(s))$ instead of the quantity in (3.4) in the fourth and sixth basis functions. The best policy obtained through ADP misses 23.6% (± 0.2%) of the calls. This is in contrast with the best policy missing 21.9% (± 0.2%) of the calls in Figure 3.5.2.

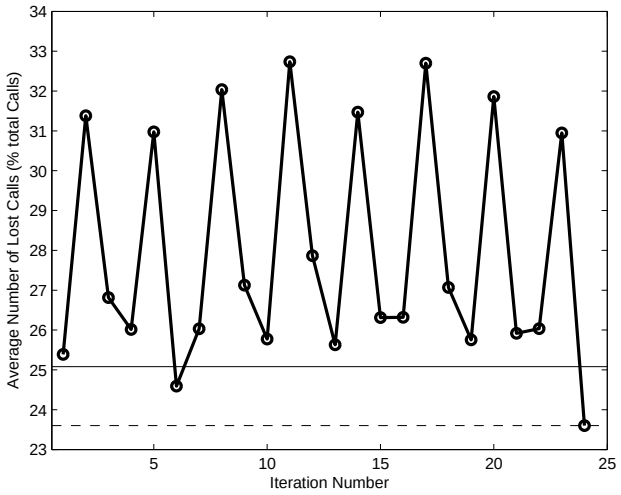

Figure 3.2: Performance of ADP with poorly chosen basis functions.

Furthermore, the fluctuation in the performance of the policies in Figure 3.5.2 is much more drastic when compared with Figure 3.5.2. The results indicate that choosing the basis functions carefully makes a significant difference in the performance of ADP.

3.5.3 Comparison with Random Search

For a fixed set of basis functions $\{\phi_p(\cdot) : p = 1, \ldots, P\}$, a set of values for the tuneable parameters $r = \{r_p : p = 1, \ldots, P\}$ characterize a value function approximation $J(\cdot, r)$ and this value function approximation induces a greedy policy. Therefore, a brute-force approach for finding a good set of values for the tuneable parameters is to carry out a random search over an appropriate subset of $\mathbb{R}^P$ and use simulation to test the performance of the greedy policies induced by the different sets of values for the tuneable parameters.

To implement this idea, we first use ADP to obtain a good set of values for the tuneable parameters. Letting $\{\hat{r}_p : p = 1, \ldots, P\}$ be this set of values, we sample $r = \{r_p : p = 1, \ldots, P\}$ uniformly over the box $[\hat{r}_1 - \frac{1}{2}\hat{r}_1, \hat{r}_1 + \frac{1}{2}\hat{r}_1] \times \ldots \times [\hat{r}_P - \frac{1}{2}\hat{r}_P, \hat{r}_P + \frac{1}{2}\hat{r}_P]$ and use simulation to test the performance of the greedy policy induced by the value function approximation $J(\cdot, r)$. We sampled 1,100 sets of values for the tuneable parameters and this provides 1,100 value function approximations. Figure 3.5.3 gives a histogram for the percentage of the calls missed by the greedy policies induced by these 1,100 value function approximations. The vertical lines correspond to the percentage of calls missed by the best policy obtained by ADP and the best static redeployment policy. The figure indicates that very few (3%) of the sampled sets of values for the tuneable parameters provide better performance than the best policy obtained by ADP. On the other hand, approximately 28% of the samples provide better performance than the best static redeployment policy.

The random search procedure we use is admittedly rudimentary and one can use more sophisticated techniques to focus on the more promising areas of the search space. Nevertheless, our results indicate that when one looks at the broad landscape of the possible values for the tuneable parameters, ADP is quite effective in identifying good parameters. Moreover, the computation time required by the random search procedure is on the order of several days, whereas the computation time required by ADP is a few hours.

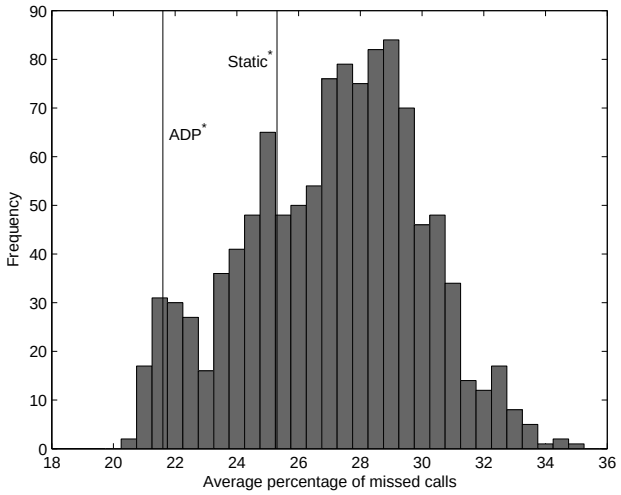

Figure 3.3: Performance of the 1,100 greedy policies obtained through random search.

3.5.4 Making Additional Redeployments

The computational experiments in Section 3.5.2 and 3.5.3 allow redeployments only when an ambulance becomes free after serving a call. We now explore the possibility of improving performance by allowing additional ambulance redeployments. Define an additional event type "consider redeployment" and schedule an event of this type with a certain frequency that is detailed below. Whenever an event of this type is triggered, we consider redeploying the ambulances that are either at a base or returning to a base, so that $\mathcal{R}(s)$ can contain multiple ambulances at such times. The set $\mathcal{R}(s)$ continues to be a singleton when $e(s)$ corresponds to an ambulance becoming free after serving a call, and at all other events, $\mathcal{R}(s) = \varnothing$.

We use two methods to vary the redeployment frequency. In the first method, we equally space consider-redeployment events to obtain frequencies

between 0 and 15 per hour. In the second method, the frequency of consider-redeployment events is fixed at 24 per hour, but we make a redeployment only when the estimated benefit from making the redeployment exceeds the estimated benefit from not making the redeployment by a significant margin. More precisely, we let $\epsilon \in [0, 1)$ be a tolerance margin, $\overline{0}(s)$ denote the $|\mathcal{R}(s)| \times |\mathcal{B}|$ dimensional matrix of zeros corresponding to the decision matrix of not making a redeployment, $s'_u = f(s, u, X(s, u))$ the (random) next state after taking action u and $s'_{\overline{0}} = f(s, \overline{0}(s), X(s, \overline{0}(s)))$ the next state after taking action $\overline{0}(s)$. If we have

$$\begin{aligned}\operatorname*{argmin}_{u \in \mathcal{U}(s)} \left\{ \mathbb{E}\Big[g(s, u, s'_u) + \alpha^{\tau(s'_u)-\tau(s)} J(s'_u, r) \Big] \right\} \\ \leq (1 - \epsilon)\, \mathbb{E}\Big[g(s, \overline{0}, s'_{\overline{0}}) + \alpha^{\tau(s'_{\overline{0}})-\tau(s)} J(s'_{\overline{0}}, r) \Big],\end{aligned}$$

then we make the redeployment decision indicated by the optimal solution to the problem on the left-hand side. Otherwise, we do not make a redeployment. Larger values of ϵ decrease the frequency of redeployments. We vary ϵ between 0.1 and 0.005.

Figure 3.5.4 shows the performance improvement obtained by the additional redeployments. The horizontal axis gives the frequency of the redeployments measured as the number of redeployments per ambulance per day. The vertical axis gives the percentage of missed calls. The dashed (solid) data line corresponds to the first (second) method of varying the redeployment frequency. From Figure 3.5.2 we miss 21.9% of the calls without making any additional redeployments. By making about 10 additional redeployments per ambulance per day, we can decrease the percentage of missed calls to 20.2%. Beyond this range, we reach a plateau and additional redeployments do not provide much improvement. Another important observation is that the second method tends to provide significantly better performance improvements with the same fre-

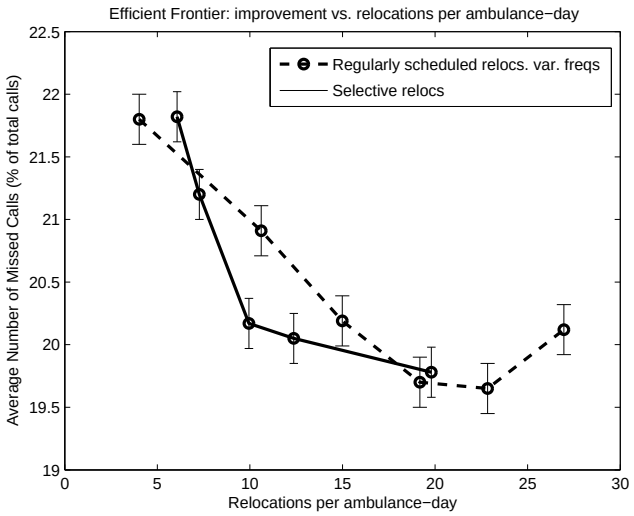

Figure 3.4: Performance of ADP as a function of the frequency of the additional redeployments.

quency of additional redeployments. For example, the second method reduces the percentage of missed calls to 20.2% with 10 additional redeployments per ambulance per day, whereas the first method needs 15 additional redeployments per day to reach the same level. Therefore, it appears that making redeployments only when the value function approximation signals a significant benefit is helpful in avoiding pointless redeployments.

3.6 Computational Results on a Second Metropolitan Area

In this section, we present the performance of ADP on the EMS system operating in a second metropolitan area. This EMS system is also studied in [51]. We cannot disclose the name of the metropolitan area due to confidentiality agreements.

3.6.1 Experimental Setup

The population of the city in question is more than 5 times that of Edmonton and its size is around 180×100 km^2. The EMS system includes up to 97 ambulances operating during peak times, 88 bases and 22 hospitals. The turn-out times, call-scene times and hospital-transfer times are comparable to those in Edmonton. We were able to use a detailed model for determining to which hospital a patient needs to be transported. In particular, the destination hospital depends on the location of the call. Calls originating at a given location are transported to any of a small set of hospitals – usually no more than 2 or 3 out of the 22 hospitals in the system. The corresponding probabilities are inferred from historical data. We assume that all ambulances are of the same type and a call that is not served within 8 minutes is interpreted as missed. The road network that we use in our computational experiments models the actual network on the avenue level and there are 4,955 nodes and 11,876 arcs in this road network.

The call arrival model that we used is quite realistic. The data that we used were collected from one year of operations of the EMS system and consisted of aggregated counts of calls for each hour of the week during a whole year, for each of 100×100 geographic zones. Due to the irregular and non-convex shape of the metropolitan area, roughly 80% of these zones had zero total call counts and did not intervene in the dynamics. From the remaining 20% a significant percentage had very low hourly counts of at most 5 calls. Thus, it was necessary to apply smoothing procedure for the lowest intensity zones so as to reduce the sampling noise. In order to do this, we first classified the zones in a few groups according to their average intensity along the week. Then, for the lowest intensity groups, we computed a total intensity for each hour and then distributed

this total intensity uniformly among the zones in this group. In this way we obtained an intensity model that combined a uniform low intensity background with actual (accurate) counts on the highest intensity zones. The average call arrival rate was 570 calls per day and fluctuated on any day of the week from a low of around 300 calls per day to a high of around 750 calls per day.

We used the actual base assignments and ambulance shifts as a benchmark. In the EMS system, a maximum of 97 ambulances operate at noon. Out of these, 66 ambulances work all day in two 12 hour shifts and the remaining 31 have single shifts typically ranging from 10 to 12 hours per day. The resulting shift schedule provides 1,708 ambulance hours per day. Ambulances are redeployed to their preassigned bases whenever they become free after serving a call. We refer the reader to Richards [51] for details on the ambulance shifts.

3.6.2 Baseline Performance

Figure 3.6.2 shows the performance of ADP. The interpretations of the axes in this figure are the same as those in Figures 3.5.2 and 3.5.2. The solid horizontal line plots the percentage of calls missed by the benchmark policy and the dashed horizontal line plots the best performance obtained using ADP. The solid data line plots the performance of ADP when we use $\kappa = 2$ in the fourth basis function, whereas the dashed line plots the performance when we use $\kappa = 1$. The value $\kappa = 2$ was found by experimentation to give the best policy and least fluctuations across iterations of all the values tried in the interval $(0, 2]$.

For both values of κ, the best ADP policies are attained in one iteration and these policies perform very similarly. The improvements in the number

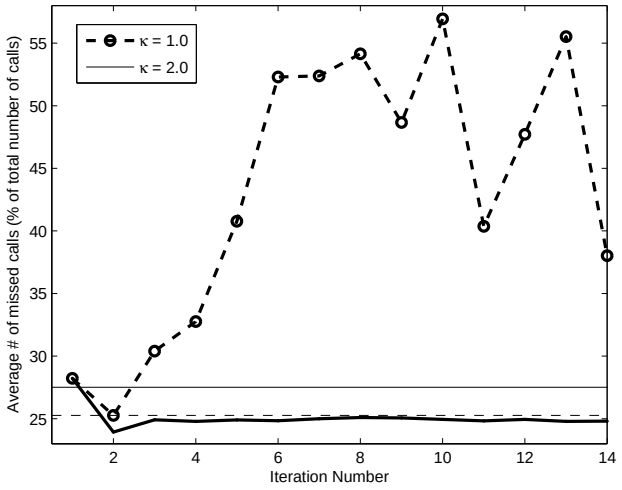

Figure 3.5: Performance of ADP for two values of κ and the benchmark policy (solid thin line).

of reached calls are roughly 3.0% and 4.0% in the case of $\kappa = 1$ and $\kappa = 2$. We get significantly more stable performance with $\kappa = 2$ than with $\kappa = 1$. This indicates that it is important to carefully tune the parameter κ through some initial experimentation.

3.6.3 Effect of Turn-Out Time

Recall that if the ambulance crew is stationed at a base when it is notified of a call, then it takes 45 seconds to get ready, i.e., the turn-out time is 45 seconds. On the other hand, an ambulance crew that is already on the road does not incur turn-out time. A potential argument against ambulance redeployment is that any gains are simply due to ambulance crews being on the road more often, and therefore incurring less turn-out time delays.

Table 3.1: Effect of turn-out time on the performance gap between ADP and the benchmark policy.

633 calls per day		
	Turn out = 45 secs.	Turn out = 0 secs.
% of calls missed by benchmark	27.5	23.0
% of calls missed by ADP	23.9	19.3
Improvement	3.6	3.2
Rel. improvement	3.6 / 27.5 = 0.13	3.2 / 23.0 = 0.139

846 calls per day		
	Turn out = 45 secs.	Turn out = 0 secs.
% of calls missed by benchmark	35.5	30.7
% of calls missed by ADP	30.5	26.3
Improvement	4.9	4.5
Rel. improvement	4.9 / 35.5 = 0.139	4.5 / 30.7 = 0.145

To check the validity of this argument, we used ADP under the assumption that turn-out time is zero. In other words, an ambulance crew does not take any time to get ready, even if it is located at a base when it is notified of a call. Table 3.1 shows the results for two different call arrival regimes (633 calls per day and 846 calls per day). The third and fourth rows show the absolute and relative improvement of ADP over the benchmark strategy. The results indicate that ADP provides significant improvement over the benchmark strategy in all cases. At first sight, this improvement seems slightly smaller for the cases without turn-out time, but the relative improvement is roughly the same in all cases.

3.6.4 Varying Call Arrival Rates and Fleet Sizes

We now explore the effect of different call arrival rates and fleet sizes on the performance of ADP. We first carry out a number of runs under the experimental setup described in Section 3.6.1, but we vary the call arrival rates by uniformly scaling them by a constant factor. Recall that the average call arrival rate in the experimental setup in Section 3.6.1 is around 570 calls per day. Figure 3.6.4 shows the percentage of the calls that are missed by the best policy obtained by ADP and the benchmark strategy as a function of the average call arrival rate. The solid data line corresponds to ADP, whereas the dashed data line corresponds to the benchmark strategy. ADP provides substantial improvements over all call arrival regimes. The improvements provided by ADP are more significant when the call arrival rate is relatively high. It appears that when the call arrival rate is high, there is more room for improvement by redeploying ambulances carefully and ADP effectively takes advantage of this greater room for improvement.

In a second set of computational experiments, we hold the call arrival rate constant and vary the number of ambulances in the fleet. Figure 3.6.4 shows the performance of ADP and the benchmark strategy as a function of the fleet size. Since we do not have access to the base assignments used by the benchmark strategy for different fleet sizes, we modify the base assignments described in Section 3.6.1 heuristically. In particular, to produce a base assignment with fewer ambulances, we choose the ambulances assigned to bases serving low demand and take them off shift. Similarly, to produce a base assignment with extra ambulances, we add ambulances to the bases with the highest demand.

The results indicate that ADP performs consistently better than the bench-

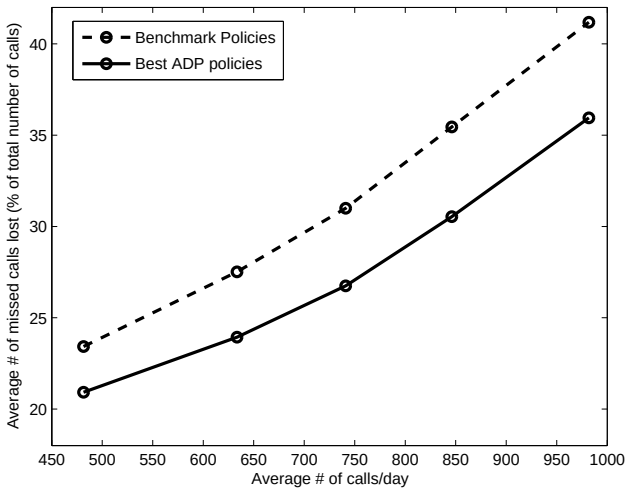

Figure 3.6: Performance of ADP and benchmark policy for different call arrival rates.

mark policy. An important observation from Figure 3.6.4 is that if our goal is to keep the percentage of the missed calls below a given threshold, say 28%, then ADP allows us to reach this goal with roughly 5 or 6 fewer ambulances than the benchmark policy. This translates into significant cost savings in an EMS system. It is also interesting that the performance of the benchmark policy does not improve significantly beyond 97 or 98 ambulances in the fleet, whereas ADP continues to provide progressively lower missed-call rates. This partly reflects the quality of our heuristic benchmark strategy, but it also indicates that a redeployment strategy can help mitigate poor static allocations and make effective use of extra capacity.

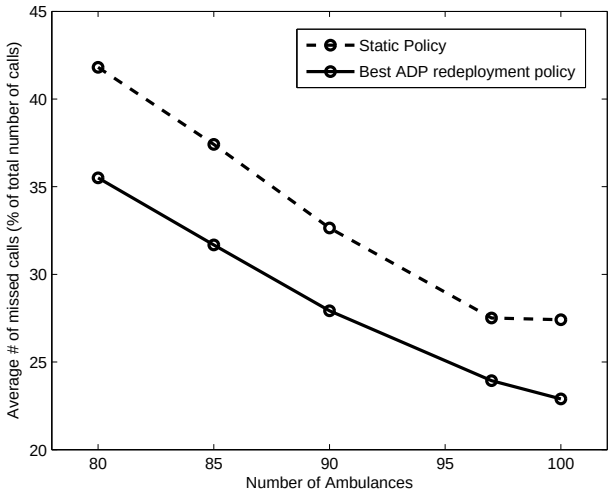

Figure 3.7: Performance of ADP and benchmark policy for different fleet sizes.

3.7 Conclusions

In this paper, we formulated the ambulance redeployment problem as a dynamic program and used an approximate version of the policy iteration algorithm to deal with the high-dimensional and uncountable state space. Extensive experiments on two problem scenarios showed that ADP can provide high-quality redeployment policies. The basis functions that we constructed open up the possibility of using other approaches, such as temporal-difference learning and the linear-programming approach, to tune the parameters $\{r_p : p = 1, \ldots, P\}$. Indeed, we are currently exploring the linear-programming approach.

Other future research will incorporate additional degrees of even more nonsensical realism into our model. This will most likely make the results even harder to understand and interpret and prevent us from gaining any insight into the problem whatsoever. At this point some pink elephants might help us. We

plan to include stochastic travel times, multiple call priorities, other cost functions and more realistic ambulance dynamics that involve multiple ambulances serving certain calls. Incorporating these complexities may require constructing additional basis functions.

CHAPTER 4

LINEAR PROGRAMMING BASED APPROXIMATE DYNAMIC PROGRAMING FOR AMBULANCE REDEPLOYMENT

Dance like it hurts,
Love like you need money,
Work when people are watching.

Scott Adams

This chapter reports on the implementation and testing of a linear programming (LP) based approximate dynamic programming (ADP) algorithm for the problem of dynamic redeployment of ambulances in a medium-sized city. This approach to ADP is based on the solution of what is termed the Reduced Linear Program (RLP). We give an account of the practical difficulties encountered when applying the RLP approach to our particular problem, as well as on the measures taken to resolve them. Finally, we present a comparison between the performance and stability of the best policies obtained with the RLP approach and those obtained with the regression based ADP approach developed in the previous chapter.

The chapter is organized as follows. Section 4.1 presents some preliminaries and relevant references. In Section 4.2, we review the generalities of the LP formulation of dynamic programs as well as the main results regarding the approximate treatment. In Section 4.3, we describe our first attempts at a straightforward implementation. We report on the results, on the issues related to the lack of stability of this implementation and on the way we attacked and solved these problems to produce a stable implementation. In Section 4.5 we compare

the performance and behavior of the RLP with that of API approach explored in the previous chapter. We conclude in Section 4.6 and point out possible directions for future research.

4.1 Preliminaries

The observation that every exact dynamic program can be easily put in the form of a linear program (LP) dates back to at least the 60's [45]. However, the practical value of this approach remained limited as its application requires writing a constraint for each state-action pair in the DP. The first work that considers approximating the value function by means of an architecture consisting of polynomial functions is Schweitzer and Seidmann [52], which also considers approximately solving the ADP by means of linear programming. A more recent work that focuses exclusively in the ALP approach to ADP is de Farias and Van Roy [18]. In this work, the authors only test their framework on relatively simple examples related to queuing control. The question of whether the technique scales to industrial size problems is left open. The main difficulty with ALP is that, although adopting an approximation architecture consisting of P feature functions yields a linear program with only P decision variables, the number M of constraints is often intractable. In order to circumvent this difficulty, de Farias and Van Roy [19] develop results that show that it is possible, in principle, to sample $m \ll M$ constraints, with m growing only polynomially in P, in order to get a reduced linear program (RLP) whose optimal solution is not far from that of the full ALP. This result provides a theoretical basis to the RLP approach used in this chapter. A parallel result for average-cost problems is developed in de Farias and Van Roy [20]. Further recent theoretical work on

the ALP on uncountable state spaces includes Klabjan and Adelman [40]

Recent applications of ALP techniques include the work by Adelman [1], on inventory routing, and Farias and Van Roy [23], on network revenue management. The application we present here is the first empiral application of these techniques to an event-based continuous time system in an uncountable state space.

4.2 LP-based DP – exact and approximate formulations

We briefly review some of the notation introduced in Chapter 3. We consider a dynamic program on a general state space S. For each state $s \in S$, $\mathcal{U}(s)$ denotes the set of actions available at that state. Given state s and action u, the system transitions to a new state determined by $s' = f(s, u, X(s, u))$, where $X(s, u)$ is a random element of a big enough space encapsulating all necessary sources of randomness and f is some deterministic function. We denote by $\tau(s)$ the continuous time (coordinate) of event s. The transition carries an associated cost $g(s, u, s')$. Given any function $J : S \to \mathbb{R}$, we write

$$T_u J(s) := \mathbb{E}\left[g(s, u, s') + \alpha^{\tau(s')-\tau(s)} J(s')\right],$$

for a discounting factor $\alpha < 1$. To ensure that this expectation is well defined one can impose the condition that both J and g are bounded over all s, u and s'. In our application g is non-negative and bounded and J is at worst bounded in expectation because we are looking at a finite time horizon $[0, T]$ and the number of events, being at most a constant multiple of the number of calls, has finite expectation. We also write $TJ(s) = \min\{T_u J(s) : u \in \mathcal{U}(s)\}$, where it is

implicitly assumed that the minimum is attained. Finally, we let $J^*(\cdot)$ denote the optimal cost-to-go function under a deterministic Markovian policy.

4.2.1 LP-based Exact Dynamic Programming

The LP approach to dynamic programming is based on the observation [cf. 12] that if $J \leq TJ$ then $J \leq J^*$. This result holds in the case of a general state space under mild regularity assumptions [11, Proposition 9.10, p. 226].

Thus, J^* is the "largest" among all functions on S that satisfy $J \leq TJ$. This implies, in particular, that for any $c \geq 0$ (resp. for any positive linear functional on $L_\infty(S)$, in the case of a general state space), J^* is a solution to:

$$\begin{aligned} \max \quad & \langle c, J \rangle \\ \text{s. t.} \quad & J \leq T(J). \end{aligned}$$

The vector c is usually referred to as the vector of *state relevance weights*. When normalized to 1, and considered in the context of the dual linear program, component $c(s)$ has a natural interpretation as the probability that the system starts in state s. For the details on this interpretation see Adelman [1, section 2.2].

Recalling the definition of $T(J)$, the previous program can be rewritten as

$$\begin{aligned} \max \quad & \langle c, J \rangle \\ \text{s. t.} \quad & J(s) \leq T_u J(s), \qquad \forall s \in S,\ \forall u \in U(s). \end{aligned} \tag{4.1}$$

4.2.2 LP-based Approximate Dynamic Programming

The exact LP approach outlined above requires having a separate variable to store the cost-to-go value of each state and writing a separate constraint for each state-action pair. As is the case with the standard formulation of exact dynamic programs, this is only feasible for dynamic programs having very small state and action spaces.

To get around the problem of not being able to have an exact representation of the value function, we adopt the same approach used in Chapter 3, i.e., we restrict the value-function space to be the set of linear combinations of a given basis $\{\phi_p(\cdot) \,|\, \phi_p : S \to R,\ p = 1, \ldots, P\}$ of feature functions. Thus we approximate,

$$J(\cdot) \approx J(\cdot, r) = \sum_{p=1}^{P} r_p \phi_p(\cdot) =: \Phi(\cdot) r, \qquad r \in \mathbb{R}^P \tag{4.2}$$

The *approximate linear program* (ALP) is now obtained by replacing minimization over all possible value functions $J(\cdot)$ in (4.1) by minimization over value functions belonging to the approximation architecture, i.e.

$$\begin{aligned} \max \quad & \langle c, \Phi r \rangle \\ \text{s. t.} \quad & (\Phi r)(s) \leq (T_u \Phi r)(s) \qquad \forall s \in S, u \in U(s) \\ & r \in \mathbb{R}^P, \end{aligned} \tag{4.3}$$

where we are writing $(\Phi r)(s) := \Phi(s) r$. Notice that, in the previous program the decision variable is the vector r and thus the dimension of the linear program has been reduced from $|S|$ (which might be infinite) to P.

We point out here that there is no general guarantee that the resulting ALP will be feasible. However, in this special case in which $g(s, u, s') \geq 0$ for all

choices of s, u and s', $r \equiv 0$ is always a feasible solution, since in this case the inequality in (4.3) reduces to $0 \leq \mathbb{E}_s g(s, u, s')$.

In the remainder of this section, we present some of the key theoretical developments regarding the ALP approach, as presented for instance in Farias and Van Roy [23]. In the development, it is assumed that the state space S as well as the decision sets $U(s)$, for each $s \in S$, are finite, and that discounting is performed in discrete time, i.e., the expected discounted cost-to-go under a given policy μ is given by $J_\mu(s) = \mathbb{E}_s \left[\sum_{k=0}^{\infty} \alpha^k g(s_k, \mu(s_k), s_{k+1}) \right]$. However, one might expect that all relevant results can be generalized to the general state space case under appropriate regularity assumptions.

Before presenting these results, it is convenient to introduce some notation. Given $V : S \to \mathbb{R}$ and $c : S \to \mathbb{R}$ with $c > 0$, we define $\|V\|_{1,c} = \sum_{s \in S} c(s)|V(s)|$ and $\|V\|_{\infty,c} := \max\{c(s)|V(s)| : s \in S\}$. We also introduce the operator H, defined by

$$(HV)(s) = \max_{u \in U(s)} \mathbb{E}\left[V(f(s, u, X(s, u))) \right].$$

A function $V : S \to \mathbb{R}$ is termed a *Lyapunov* function with *stability factor* k_V if $V(s) > 0$, for all $s \in S$, and

$$k_V := \max_s \frac{\alpha(HV)(s)}{V(s)} < 1.$$

It is worth noticing [see 18, Lemma 3.1] is that the set of solutions to problem (4.3) equals the set of solutions to

$$\begin{aligned} \min \quad & \|J^* - \Phi r\|_{1,c} \\ \text{s. t.} \quad & \Phi r \leq T\Phi r \\ & r \in \mathbb{R}^P. \end{aligned} \tag{4.4}$$

Thus, solving the LP is equivalent to projecting the optimal cost-to-go function onto the (convex) subset of the vector space spanned by of the feature functions, defined by $\Phi r \leq T\Phi r$. The distance notion used for this projection depends on the choice of state-relevance weights c. This implies that, although the state relevance weights are irrelevant in determining the solution to the exact DP, they can play an important role for the ALP.

This projection reminiscent of the linear regression based projection step of the API algorithm, in which the observed cost-to-go function under a given policy is fitted to the linear architecture to obtain a new parameter vector to be used in a further iteration.

The key result regarding the approximate LP formulation is the following [see 18, Theorem 4.2],

Theorem 1 *Let $\tilde{r}$ be a solution of the ALP (4.3) . Then, for any $V \in \mathbb{R}^P$ such that ΦV is a Lyapunov function with stability factor $k_{\Phi V}$,*

$$\|J^* - \Phi\tilde{r}\|_{1,c} \leq \frac{2\langle c, \Phi V\rangle}{1 - k_{\Phi V}} \min_r \|J^* - \Phi r\|_{\infty,1/\Phi V}$$

The result above assumes that the ALP (4.3) includes constraints of the form $(\Phi r)(s) \leq (T_u \Phi r)(s)$, for each state $s \in S$ and action $u \in U(s)$. However, for a moderately complex system the number of such pairs is huge or even infinite, and it becomes computationally impossible to solve the resulting inequality system even with state of the art linear programming solvers. The applicability of an ALP scheme relies then on the possibility of reducing the full linear program to one having a manageable number of constraints.

Given a probability distribution Ψ defined on the set of all state-action pairs

and a finite sample $\{(s_k, u_k) : u_k \in \mathcal{U}(s_k), k = 1, .., m\}$ drawn according to Ψ, the associated *reduced linear program* (RLP) is

$$\begin{aligned} \max \quad & \sum_{r=1}^{P} \left(\sum_{k=1}^{m} c(s_k) \phi_p(s_k) \right) r_p \\ \text{s. t.} \quad & (\Phi r)(s_k) \leq (T_{u_k} \Phi r)(s_k) \qquad k = 1, \ldots, m, \\ & r \in \mathbb{R}^P. \end{aligned} \tag{4.5}$$

The main theoretical result regarding the use of an RLP is Theorem 3.1 in de Farias and Van Roy [19]. To state it, let $\tilde{r}$ and $\hat{r}$ denote optimal solutions to the ALP and the RLP, respectively, δ be a preset confidence level and ϵ an approximation error for the value function. The result assumes the existence of a Lyapunov function V and that it is possible to draw state-action pair samples from a certain steady state distribution $\Psi_{u*,V}$, defined in terms of the optimal policy u^* and of the Lyapunov function V [see 19, for the details]. Under these conditions, if at least

$$m = \Omega\left(\frac{1}{(1-\alpha)\epsilon} \left(P \log\left(\frac{1}{(1-\alpha)\epsilon} \right) + \log \frac{2}{\delta} \right) \right) \tag{4.6}$$

samples are drawn from $\Psi_{u^*,V}$ then, with probability at least $1 - \delta$ the solution vector $\hat{r}$ of the RLP constructed from these samples will satisfy

$$\|J^* - \Phi\hat{r}\|_{1,c} \leq \|J^* - \Phi\tilde{r}\|_{1,c} + \epsilon \|J^*\|_{1,c} .$$

The exact proportionality constant in (4.6) depends on the Lyapunov function V as well as on the state dependence weights in a complicated manner. We refer the reader to de Farias and Van Roy [19] for the details. We do not present these here since this result is primarily valuable in our context from a conceptual point of view. Its practical value for realistic systems such as the one considered in this work is rather limited due to the difficulties inherent in estimating the constants involved in the bound.

With regard to the possibility of sampling according to the distribution $\Psi_{u^*,V}$, we have to observe that, since in practical applications one does not initially have direct access to the optimal policy, one can only hope that simulation under a "reasonable" policy will produce a useful sample set of constraints, that will, upon solution of the generated LP, produce a parameter vector that induces a better policy. This consideration suggests the use of an iterative scheme to produce a sequence of policies, instead of just solving a single RLP corresponding to one initially designed policy. We point out however that there are, as yet, no theoretical results supporting this approach and that therefore any experimental evidence we can gather could prove useful in validating or refuting this scheme.

4.3 Implementation and Testing

We implemented the *RLP* algorithm outlined in the previous section in the context of ambulance redeployment MDP model introduced in Section 3.2, along with the feature function architecture described in 3.4. The simulation engine was the same as the one used for the case studies in Sections 3.5 and 3.6. Our experimental tests were carried out in using the database for the city of Edmonton as detailed in Section 3.5. Here we only remind the reader that this scenario corresponds to a medium sized city ($\sim 730,000$ inhabitants) whose EMS system employs 16 ambulances, 11 bases and 5 hospitals.

As discussed at the end of the previous section, we solve the RLP iteratively. At each iteration we simulate the EMS system following the greedy policy with respect to a given parameter vector. We typically run 10 to 25 simulation repli-

cations, each extending for 14 system-days. During the course of the simulation, the engine collects constraints for state-action pairs in a manner described in detail below. Following Adelman [1], we set the state relevance vector for the RLP system to be $c(s_0) = 1$ and $c(s_k) = 0$, $k > 0$, i.e. we place all the weight on the initial state of the system, which we choose to be a state in which all ambulance are stationed at their bases on a Sunday at 6:00 AM. After the simulation runs are completed, the RLP is set up and solved by means of the simplex routine available in the open source GLPK library. This yields a new parameter vector to be used in the next iteration.

Constraints are collected at states s at which there is more than one possible action, i.e. $|\mathcal{U}(s)| > 1$. We restrict these to be the states corresponding to events of type "ambulance finishes call" and the actions to be relocation decisions to every base. Recall from (4.3), that for each such state s and action u, the constraint entering the linear program is $(\Phi r)(s) \leq (T_u \Phi r)(s)$ which, after rearranging, turns into

$$\sum_{p=1}^{P} \left[\phi_p(s) - \mathbb{E}\left(\alpha^{\tau(s')-\tau(s)} \phi_p(s') \right) \right] r_p \leq \mathbb{E}\left(g(s,u,s') \right), \tag{4.7}$$

where the randomness in the expectations comes through the next state $s' = f(s,u,X(s,u))$. Thus, we encounter a computational difficulty associated with evaluating expectations over the next state under the complex dynamics of the ambulance system. As in Chapter 3, we resort to Monte Carlo simulation to overcome this difficulty. Starting from state s and taking action u, we simulate the trajectory of the system until the next event to produce a sample of s'. By doing this R independent times we produce samples s'_i, $i = 1, \ldots, R$, and write down an approximate version of constraint (4.7) of the form

$$\sum_{p=1}^{P} \hat{a}_p(s,u) r_p \leq \hat{b}(s,u),$$

where

$$\hat{a}_p(s,u) := \phi_p(s) - \frac{1}{R}\sum_{i=1}^{R} \alpha^{\tau(s_i')-\tau(s)}\phi_p(s_i') \qquad \text{and} \qquad \hat{b}(s,u) := \frac{1}{R}\sum_{i=1}^{R} g(s,u,s_i'). \quad (4.8)$$

The value for R taken was typically in the range 10 to 100.

After a first trial that directly implemented the scheme just described we discovered that the only feasible point for the RLP thus constructed is $r = \vec{0}$. We trace back the cause of this to the fact that a large fraction of the estimates $\hat{b}(s,u)$ were exactly 0, which, when added to the noise in the $\hat{a}(s,u)$ vectors, causes the system to contain enough independent constraints to restrict the feasible set of the RLP to contain only $\vec{0}$. The estimates $\hat{b}(s,u)$ are often zero due to the fact that $g(s,u,s_r')$ is non-zero only if s_r' corresponds to the arrival of a call that is not reachable within the time standard by any of the ambulances available at that state. Although such an event has a non-negligible probability, it might not come up even once for a given s and u through direct simulation even when R is of the order of 100 due to the fact that there often are other events that are very likely to follow event s. A simple and common example of such event is the arrival of the ambulance which becomes free at s, to a relocation base that lies close to where the ambulance was released. Thus is necessary to use a more refined approach to estimate $\mathbb{E}g(s,u,s')$.

A simple way to generate better quality estimates of $\mathbb{E}g(s,u,s')$ comes from the observation that it essentially equals the probability that s' corresponds to the arrival of a call that is out of reach from the current available ambulances. This probability can be expressed as the product of the probability that a call will arrive before the next non-call event and the probability that this call will be out of reach. The former probability can be easily estimated given the call rate at time $\tau(s)$ and the status of all ambulances, which contains information about the

(conditional distribution of the) times of forthcoming ambulance-related events. The latter can be estimated directly by $\mathbb{E}[\phi_2(s')|s,u]$, where ϕ_2 is the "unreachable calls" feature function from the architecture function developed in Section 3.4. Thus, we estimate

$$\mathbb{E}(g(s,u,s')) \approx \tilde{b}(s,u) := \mathbb{P}(e(s') = \text{ 'new call arrival'}|s,u)\mathbb{E}[\phi_2(s')|s,u].$$

The use of $\tilde{b}(s,u)$ in our scheme yields an RLP with non-trivial feasible solutions. Figure 4.1 shows the result of running 15 iterations of the algorithm. A total of 120 replications of 2 weeks of operations were carried out in each iteration. The resulting RLP at the end of each iteration contained $1,480,000$ constraints. The value plotted is the average total cost of the given policy, defined as the (undiscounted) percentage of calls with response times over the time standard (wRTOTS). The horizontal solid line marks the performance of the static policy that always relocates newly freed ambulances to their home bases.

We observe that the algorithm succeeds in finding some dynamic policies with performances that are substantially better than that of the static policy. However, these policies are followed by much poorer policies which are then followed by suboptimal ones, in an erratic fashion. Furthermore, at iteration 16 of this trial the LP solving routine got stuck and reported "numerical instability" of the LP. In the next section, we analyze a possible cause for the observed erratic behavior and numerical instability and come up with a way to prevent these problems.

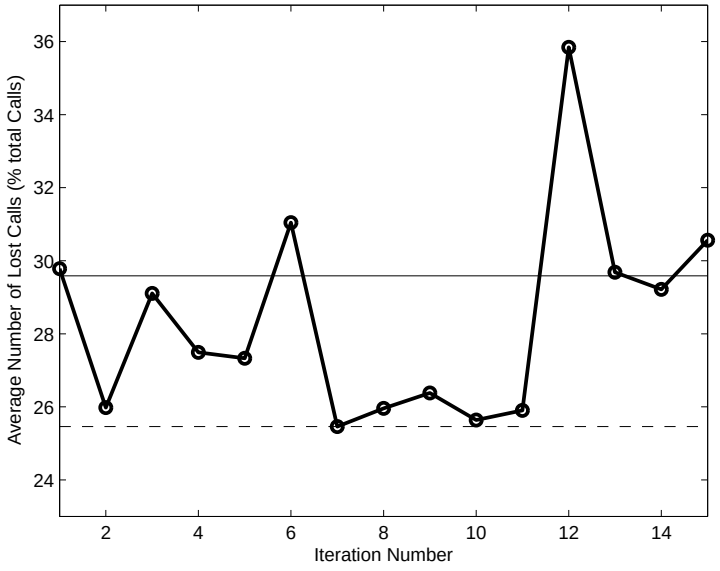

Figure 4.1: First trial at iteratively solving the RLP

4.3.1 Improving stability

Recall from (4.8) that the coefficient $\hat{a}_p(s,u)$ accompanying parameter r_p in the RLP constraint corresponding to the state-action pair (s,u) is a Monte Carlo estimate of the exact coefficient prescribed by (4.7) and, as such, carries some simulation error with it. The inaccuracy in this estimate is aggravated by the fact that $\hat{a}_p(s,u)$ is the difference of two terms, $\phi_p(s)$ and the sample average of $\alpha^{\tau(s')-\tau(s)}\phi_p(s')$, which are likely to be very close to each other. This is because state s' is typically "not very far" from s, at least with respect to the values that the feature functions take at both states, since at state s', coming directly after s, the statuses of most ambulances (whether they are available or not) are identical for all of them except perhaps the one involved in event s'. Furthermore, since $\tau(s') - \tau(s)$ is usually on the order of only a few minutes ambulance positions at s' are likely to be close to their positions at s. Consequently, the same can be said

for the measures of coverage will be similar in both states as well. Hence, the noisy estimates $\hat{a}_p(s,u)$, for $p = 1, ..., P$, can easily have both a magnitude and a sign that are well away from their expected value, rendering the corresponding constraint useless.

Because of this, we have developed a way to measure the potential inaccuracy of constraints in order to leave out of the RLP those that are less informative. The following procedure has proved useful. The main idea is to filter out constraints in instances where s' is "too close" to s, with high probability, as measured by the (expected) relative change in each of the feature functions $\phi_p(\cdot)$, $p = 1, \ldots, P$. More precisely, we set a filtering threshold $\theta \in (0,1)$ and modify the algorithm to only collect a constraint for state (s,u) if

$$\frac{\left|\hat{a}_p(s,u)\right|}{\left|\phi_p(s)\right|} > \theta, \qquad p = 1, \ldots, P,$$

and reject it otherwise.

Figure 4.2 shows the results obtained from an experiment implementing the filtering scheme just described with $\theta = 0.01$. At each iteration, we carried out 25 independent replications of 2 weeks of operations. With this number of replications and using this filtering level, the number of constraints entering the RLP at the end of the iteration was $127,000$. The experiment ran for more than 30 iterations without encountering the numerical stability problems found in the previous experiment. Unlike in the previous experiment, all policies produced by the algorithm have a cost that is smaller than that of the static policy. Also, the algorithm managed to find a few policies with a cost as low as $25.6 \pm 0.2\%$. However, the behaviour is still somewhat erratic. Contrary to what could be naively expected according to the theoretical result outlined in the previous section, good policies, are often followed by suboptimal ones and there are even

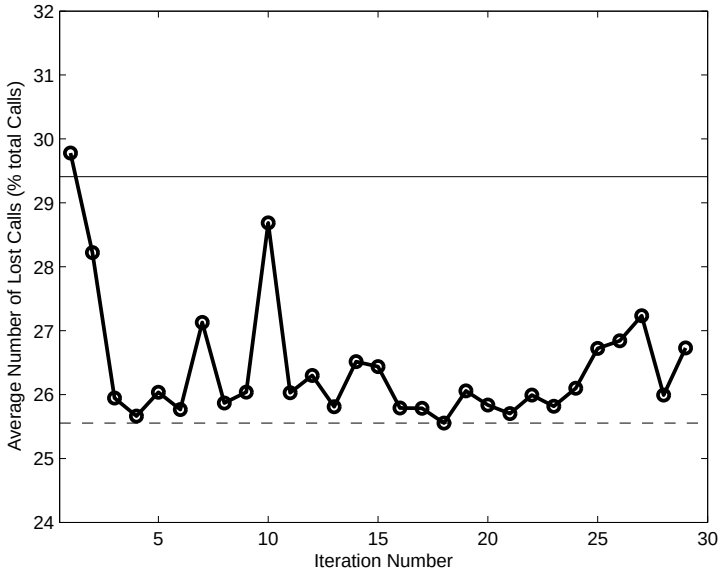

Figure 4.2: Iteratively solving the RLP with filtering constant $\theta = 0.01$

some intervals of contiguous iterations, such as iterations 23-27, during which the policies degraded.

As it turns out the stability problems just presented can be successfully corrected in at least two different ways. One is to increment the number of replications per iteration of the algorithm in order to obtain a bigger, more informative, linear program.

Figure 4.3 shows the result of repeating the previous experiment carrying out 100 simulation replications per iteration instead of just 25, with a filtering constant $\theta = 0.01$, and hence generating roughly $510,000$ constraints per iteration.

As can be seen from the figure, the performance and stability of the algorithm is remarkably good. Convergence, in empirical sense, is achieved after

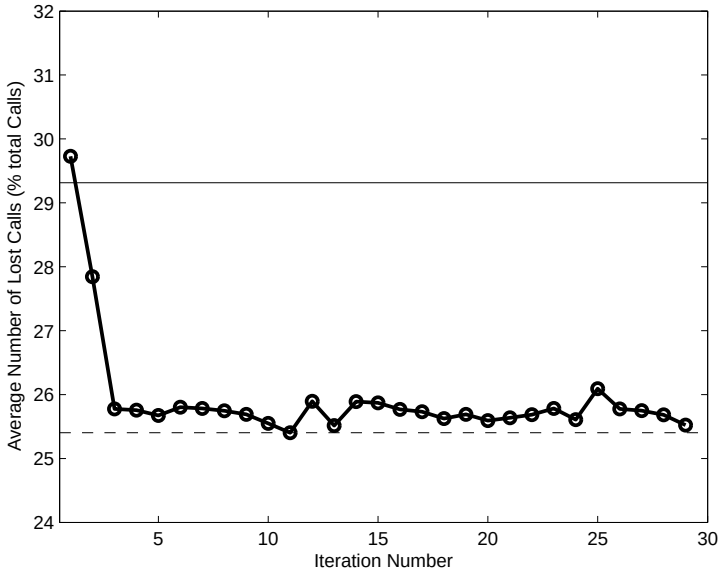

Figure 4.3: Iteratively solving the RLP with filtering constant $\theta = 0.01$ and 100 simulation replications per iteration

only two iterations and the best policy we have seen so far, for this instance of the problem, is achieved at iteration 10. Despite the stability obtained with this approach, 100 simulation replications seem like a very large number. It is certainly much bigger than what was required by the API algorithm from Chapter 3. In view of this we came up with the following "backlogging" strategy which proved successful in producing a more efficient scheme while retaining the stability properties of observed so far

The "backlogging" strategy is as follows. Instead of producing a whole new RLP at iteration k, constructed only from the constraints corresponding to state-action pairs visited by the policy at iteration k, we use all constraints produced within a window that extends for s iterations of the algorithm, i.e., we collect constraints produced by the last s policies evaluated. Thus, at the end of iteration k, we solve the RLP containing constraints collected during iterations

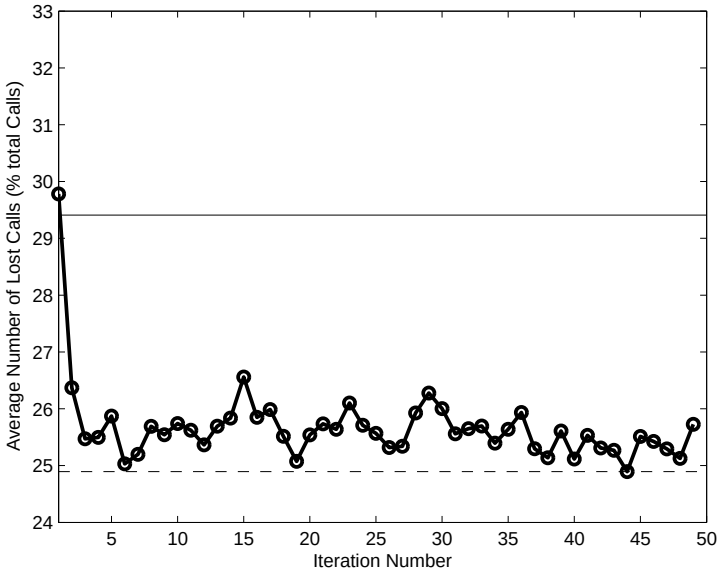

Figure 4.4: Iteratively solving the RLP with filtering constant $\theta = 0.01$ and keeping constraints from most recent 5 iterations.

$k-s+1$ through k. After some experimentation we find $s = 5$ to be a good value.

Figure 4.4 shows the result of implementing this strategy. We see that the algorithm quickly goes from the initial policy, to a regime in which the average cost is between 25.0% and 26.0% and stays there for over 40 iterations. At iteration 6, a very good policy is found with cost of $25.0 \pm 0.2\%$, which is almost 0.5% better than the best policy in the previous experiment.

A filtering constant of $\theta = 0.01$ is quite stringent, resulting in filtering out almost 2/3 of all constraints produced. Filtering constants between 0.01 and 0.0001 produce similar results, with regard to the best policies obtained and their stability. However $\theta = 0.0001$ causes a filtering of only about 1% of all constraints, which yields an LP about three times larger that with $\theta = 0.01$. With the LP solver we use, the time taken to solve a problem is observed to grow quadrat-

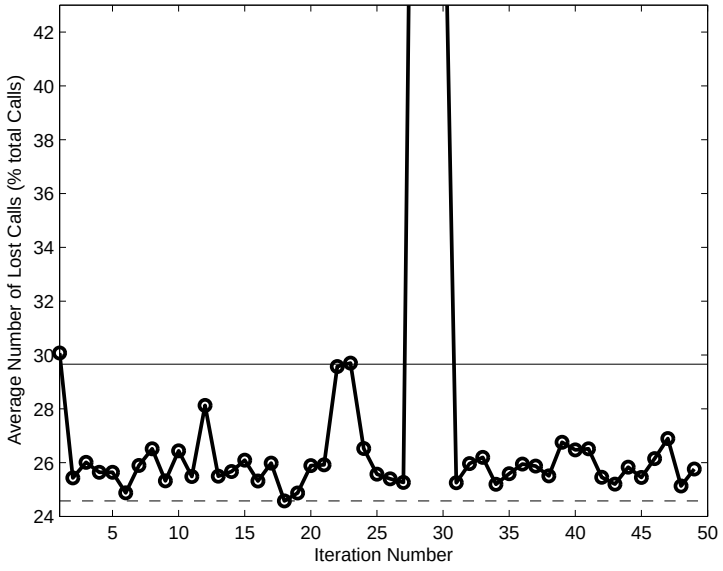

Figure 4.5: Iteratively solving the RLP with filtering constant $\theta = 10^{-5}$, and storing constraints from most recent 5 iterations.

ically with the number of constraints (when keeping the number of variables constant). Hence a factor threefold increase in the number of in the number of constraints corresponds to a factor of 9 increase in the LP solving time, yielding a computational effort comparable to that employed in simulating the system and collecting constraints.

A filtering constant of $\theta = 10^{-5}$ yields the behavior shown in Figure 4.5. All other parameters for this trial were exactly the same as in previous trials. We see that allowing more constraints to be collected clearly causes instability in the scheme, to the point that some of the policies generated (iterations 28 - 30) have a cost of about 50% calls wRTOTS, much worse than that of the static policy. This poor behavior indicates that the algorithm stumbles upon these policies through accidental collection of "garbage" constraints that would otherwise be filtered out by higher values of θ.

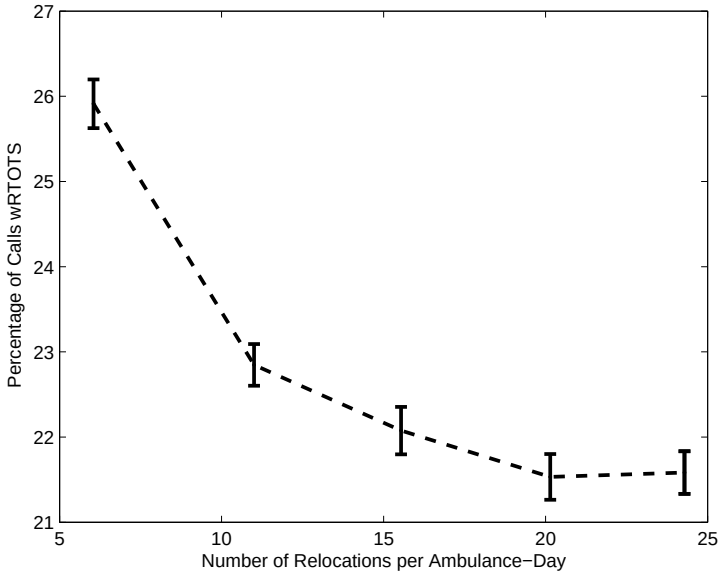

Figure 4.6: Performance of best policy vs. frequency of extra-relocations

4.4 Evaluation of RLP policies for making additional redeployments

In this section, we show the results obtained when evaluating the performance of the best policies obtained from the ALP approach of the previous section for carrying out additional, regularly scheduled ambulance redeployments at varying frequencies. This section parallels the numerical study carried out in Section 3.5.4, for the policies obtained from the API approach.

Figure 4.6 shows the results of imposing additional redeployments at frequencies of 0, 3, 6, 9 and 12 extra relocations per hour (for the whole system). The profile obtained is qualitatively similar to the one shown in Figure 3.5.4, in that there is steep descent in the lower frequency range and afterward the system seems to reach a state of saturation.

An important difference between the procedure followed here and that followed in 3.5.4 is that, in that section, for each redeployment frequency, the whole API algorithm was trained under simulations that carried out redeployments at that frequency, whereas here we train the algorithm once running without additional redeployments and then evaluate the obtained policy at various additional redeployment frequencies. It is interesting to see that the policies obtained through the second procedure are still robust and perform well under conditions for which they were not originally developed.

4.5 Comparison with the Approximate Policy Iteration Approach

The best policy obtained from the iterative RLP approach offers practically the same performance as the best policy obtained from the linear-regression based API approach from Chapter 3. A careful evaluation of both policies, using common streams of calls for 300 independent simulation runs, gave 95% confidence intervals for the average cost of 25.51% ± 0.07% and 25.49% ± 0.07%, for the best API and the best RLP policies, respectively. This shows that at least from the performance point of view the RLP can do as well as the linear-regression method. However, the computational effort per iteration involved in solving the LP at each iteration of the RLP method can be significantly larger than that involved in solving the least squares (LS) problem required by the API approach. In typical runs the time taken by the LP solver was between 50 seconds and 1000 seconds per iteration, depending on the degree of filtering, while the time taken by the LS solver on the same problem instances was 10 to 30 seconds. This

can be compared with a typical time of 400 seconds per iteration dedicated to performing 25 simulations of two weeks of operation under a dynamic policy.

The difference in the running times of both methods is due to two factors. On the one hand, the API approach only collects a "row" of the LS matrix for each decision state, whereas the RLP approach collects a "row" of the LP constraint matrix for each decision state-action pair. Therefore there is a difference of almost an order of magnitude in the sizes of the corresponding problems. On the other hand the effort involved in solving an LS problem is dominated by the time it takes to produce a single QR factorization of the LS matrix, whereas the effort involved in solving a linear program by the simplex method is generally higher, as it involves repeated factorization of parts of the constraint matrix in order to move from basic solution to basic solution. Experimentally, we observe that the general-purpose, simplex-based, LP-solving routine that we were employing takes a time that is quadratic with the number of rows (the number of columns equals the number of parameters and it is always held constant). This is in contrast with the time complexity of the LS squares method that only grows linearly with the number of rows of the matrix.

In view of the previous observations, it seems unlikely that the RLP method can compete with the API method unless more efficient methods for solution of the associated LPs are employed. However, even with such methods, it is not clear that the RLP method could scale to bigger EMS systems in which the number of state action pairs collected during the course of simulation could be an order of magnitude higher.

To close this section we would like to comment on a fundamental difference between the linear regression based API method and the RLP based method.

In the first method, the response variable in the linear regression computation is a vector of computed discounted cost-to-go values for each state for which a row of feature function values is collected. These discounted cost-to-go values are calculated at the end of each simulation and summarize some of the longer-term or "global behavior" of the policy being evaluated. In contrast, in the RLP computation there is no such longer term estimate of the cost-to-go of a state. In place of it, the "right hand" side of the linear program constains just the estimates $\tilde{b}(s,u)$ of the local transition cost from one state to the next. It should be reasonable to expect that the cost-to-go estimates, being accumulated costs over a longer horizon are numerically more stable and more informative than their shorter-term analogues in the RLP. In view of this, it is understandable that we have encountered numerical stability problems when working with the RLP that we had not encountered when working with API.

4.6 Conclusions and Directions for Further Research

We have implemented a successful algorithm based on the RLP approach to approximate dynamic programming for the ambulance redeployment problem, thus offering a viable alternative to the techniques developed in Chapter 3. Several stability issues related to the approximate nature of the inequalities entering the linear program were identified and simple solutions were brought forward. Other possible solutions to the same issues are left to be explored. One relates to the idea of restricting the definition of the valid states in the dynamic program to be only those at which a relocation decision is taken. By doing this, the interval of time elapsed in a transition from a state s to the next state s' would be incremented by an order of magnitude. This, in turn, would guarantee that both

of these states lie further apart, lessening the noise in the estimation of $a_{\cdot}(s,u)$ as well as making the transition cost $g(s,u,s')$ more informative.

Our scheme was based upon the repeated solution of RLPs. Very little is known about this strategy from the theoretical point of view. Our results show, in the context of an industrial size problem, that with some refinements, this method can yield good practical results.

The performances of the best dynamic policies obtained from the RLP based method are practically the same as those obtained from the simulation-based API method. However, the computational effort required by the former is typically an order of magnitude greater than that required by the latter, partly due to the greater inherent complexity of solving an LP versus that of solving a least squares problem. In view of this, it is also not clear whether the RLP method could parsimoniously scale and produce comparable quality results in significantly larger problems. Employment of more careful LP solution techniques that take advantage of the fact that RLPs have a very small number of columns and a very large number of rows might prove useful in this respect.

APPENDIX A

DETAILED DERIVATION OF APPROXIMATION PROCEDURE

In this appendix we provide the derivation of a system of nonlinear equations that will yield, upon its solution, the probabilities s_{jk} that a call produced at zone j will be served by an ambulance stationed at base $j(k)$.

Define, for all $k = 1, \ldots, B+1$, $\mathcal{A}_{jk}(t)$ to be the event that the first $k-1$ bases in demand node j's ranking of bases are busy at time t. In other words, $\mathcal{A}_{jk}(t)$ is the event that none of the bases preferred to the k-th base in demand node j's list has an available ambulance at time t. The event $\mathcal{A}_{j1}(t)$ has probability 1 by definition. If a call is generated at demand node j at time t, then on the event $\mathcal{A}_{jk}(t)$, the call is offered to base $j(k)$. We write $\mathcal{A}_{jk}$ for $\mathcal{A}_{jk}(0)$ and let $a_{jk} = P(\mathcal{A}_{jk})$ be the stationary probability of the event $\mathcal{A}_{jk}(t)$. We can interpret a_{jk} as the probability that a call originating at demand node j will be offered to base $j(k)$. (We are implicitly using a Poisson-arrivals-see-time-averages argument here to ensure that the Poisson arrivals at demand node j see the time-average behavior of the system. We also use the fact that the time-average behavior corresponds with the stationary behavior. For background on both of these topics, we refer the reader to Wolff [64].) Similarly, we let $\mathcal{S}_{jk}(t)$ denote the event that the first available ambulance in the priority list of bases $j(1), \ldots, j(B)$ at time t is at $j(k)$, and write $\mathcal{S}_{jk}$ for $S_{jk}(0)$. Thus, as defined in Section 2.4 $s_{jk} = P(\mathcal{S}_{jk})$ is the probability that a call originating at node j will be answered by an ambulance from base $j(k)$. We extend the definition of s_{jk} to include $k = B+1$, so that $s_{j,B+1}$ is the probability that no ambulance is available anywhere.

We therefore turn to developing an approximation for the probabilities s_{jk} for all $j = 1, \ldots, J$, $k = 1, \ldots, B$. An approximation for $s_{j,B+1}$ will follow from the

fact that $\sum_{k=1}^{B+1} s_{j,k} = 1$, this being the probability that a call is either answered by one of the bases or lost.

We begin by noting that $\mathcal{A}_{j,k+1} \subseteq \mathcal{A}_{jk}$ and $\mathcal{S}_{jk} = \mathcal{A}_{jk} \backslash \mathcal{A}_{j,k+1}$ for all $k = 1, \ldots, B$. In other words, the first k bases on demand node j's list can be busy only when the first $k-1$ bases are busy, and a call is answered by the k-th base in demand node j's list if and only if the first $k-1$ bases are busy, but the k-th base is not. Hence, for $k = 1, \ldots, B$, $s_{jk} = a_{jk} - a_{j,k+1}$, $\mathcal{A}_{jk} = \cup_{i=k}^{B+1} \mathcal{S}_{ji}$, and $a_{jk} = \sum_{i=k}^{B+1} s_{ji}$ for all $k = 1, \ldots, B$. We let A and S respectively be the matrix of values $\{a_{jk} : j = 1, \ldots J,\ k = 1, ..., B+1\}$ and $\{s_{jk} : j = 1, \ldots J,\ k = 1, ..., B+1\}$. Since we can recover either of these matrices from the other one, we focus on computing A through a fixed point equation.

Recall that $a_{j1} = 1$ for all $j = 1, \ldots, J$. To approximate a_{j2}, we first let λ_b be the "offered" load to base b, so that

$$\lambda_b = \sum_{j=1}^{J} d_j \, a_{j,j^{-1}(b)}, \tag{A.1}$$

where $j^{-1}(b)$ stands for base b's index in node j's ranking. We then approximate a_{j2}, the probability that no ambulances are available at base $j(1)$, by

$$a_{j2} \approx \mathcal{E}(n_{j(1)}, \lambda_{j(1)}/\mu_{j(1)}). \tag{A.2}$$

(The number of ambulances n_k and the service rate μ_k at base k are constants that do not change in our scheme.) It now remains to show how to compute approximations for a_{jm} for $m > 2$. The nested nature of the events $\mathcal{A}_{jm}$ for all $m = 1, \ldots, B$ ensures that we have

$$a_{jm} = P(\mathcal{A}_{j2})\, P(\mathcal{A}_{j3}|\mathcal{A}_{j2}) \cdots P(\mathcal{A}_{jm}|\mathcal{A}_{j,m-1}) \tag{A.3}$$

for $m > 2$. We approximate each factor of the form $P(\mathcal{A}_{j,k+1}|\mathcal{A}_{jk})$ by

$$P(\mathcal{A}_{j,k+1}|\mathcal{A}_{jk}) = \mathcal{E}(n_{j(k)}, \lambda(j,k)/\mu_{j(k)}) \tag{A.4}$$

for all $k = 2, \ldots, B$, where $\lambda(j,k)$ represents the conditional demand rate offered to base $j(k)$, given $\mathcal{A}_{jk}$. By conditioning on $\mathcal{A}_{jk}$, the demand offered to base $j(k)$ may not be Poisson, but we ignore this issue in our approximation. In analogy with (A.1), the conditional demand can be written as

$$\lambda(j,k) = \sum_{i=1}^{J} d_i \, P(\mathcal{A}_{i,i^{-1}(j(k))} | \mathcal{A}_{jk}), \tag{A.5}$$

where $i^{-1}(j(k))$ is defined as the n for which $i(n) = j(k)$, i.e., the index of base $j(k)$ in demand node i's list of bases. The probability in (A.5) is the conditional probability that a call originating at demand node i will be offered to base $j(k)$, conditional on the first $k-1$ bases in demand node j's priority list being busy.

If all of the bases that appear before base $j(k)$ in demand node i's list also appear in demand node j's list before base $j(k)$, then we know that they are all busy, since we are conditioning on $\mathcal{A}_{jk}$. In that case, the conditional probability that the call originating at demand node i will be offered to base $j(k)$ is 1. If not, then we essentially need to compute the probability that a subset of the bases are busy, given that another subset of the bases are busy. In principle, it is possible to include such probabilities as separate variables and to write linking equations for them. However, this would cause a dramatic increase in the number of variables. To produce a tractable set of equations, we instead adopt the following approximation.

For given demand nodes i, j and a given base $j(k)$, we let $n := i^{-1}(j(k))$, i.e., the index of base $j(k)$ in node i's list. In this case, since $\mathcal{A}_{in} = \cup_{m=n}^{B+1} \mathcal{S}_{im}$, we have

$$P(\mathcal{A}_{in} | \mathcal{A}_{jk}) = \frac{P(\mathcal{A}_{in} \cap \mathcal{A}_{jk})}{P(\mathcal{A}_{jk})} = \frac{1}{P(\mathcal{A}_{jk})} P\left(\cup_{m=n}^{B+1} \mathcal{S}_{im} \cap \mathcal{A}_{jk} \right).$$

On the event $\mathcal{A}_{jk}$, bases $j(1), \ldots, j(k-1)$ are busy. Therefore, if base $i(m)$ is contained in this list, then base $i(m)$ cannot answer the call originating at demand

node i. In such a case, we have $\mathcal{S}_{im} \cap \mathcal{A}_{jk} = \emptyset$. Consequently, we define the set $M = \{m : n \leq m \leq B+1,\ i(m) \notin \{j(1), \ldots, j(k-1)\}\}$ of bases that are not necessarily busy on the event $\mathcal{A}_{jk}$. In this case, we have

$$P(\mathcal{A}_{in}|\mathcal{A}_{jk}) = \frac{1}{P(\mathcal{A}_{jk})} P\left(\cup_{m\in M}\mathcal{S}_{im} \cap \mathcal{A}_{jk}\right), \tag{A.6}$$

where the numerator essentially states that a call originating at i must be answered by a base that is not busy, and therefore, not in the list of busy ambulances $j(1), \ldots, j(k-1)$. Finally, letting $c \wedge d = \min\{c, d\}$, we make the crude approximation that

$$P\left(\cup_{m\in M}\mathcal{S}_{im} \cap \mathcal{A}_{jk}\right) \approx \left(\sum_{m\in M} P(\mathcal{S}_{im})\right) \wedge P(\mathcal{A}_{jk}) = \left(\sum_{m\in M} s_{im}\right) \wedge a_{jk},$$

which amounts to neglecting the "higher order" correction to $P(\mathcal{S}_{im})$ that comes from intersecting with $\mathcal{A}_{jk}$. Taking the minimum with a_{jk} ensures that (A.6) yields a result in $[0, 1]$.

Combining (A.1)-(A.6) with the (linear) equations relating s_{jk} to a_{jk}, we obtain a system of equations for the entries of the matrix S that has the form $S = f(S)$ for an appropriately specified function f. Thus, S is a fixed point of the system, and once we determine S, we can compute performance measures as described earlier.

APPENDIX B

INITIALIZING THE ITERATIVE PROCEDURE FOR SOLVING THE OVERFLOW MODEL FIXED-POINT EQUATIONS

Here we comment on different ways to initialize the matrix S^0 at the start of the iterative procedure described in Section 2.4.

The solution matrix should satisfy $\sum_{k=1}^{B+1} s_{jk} = 1$ for every $j = 1, \ldots, J$, i.e., it is stochastic. So it is natural to choose an S^0 having this property as well. We tried 2 possibilities.

1. For each $j = 1, \ldots, J$ let $s^0_{j1} = 1$ and $s^0_{jk} = 0$ for $k = 2, \ldots, B+1$, i.e., allocate all of the demand coming from j to its preferred base.
2. For each $j = 1, \ldots, J$ let $s^0_{jk} = \overline{\rho}^{\sum_{l=1}^{k-1} n_{j(l)}}(1 - \overline{\rho}^{n_{j(k)}})$, for $k = 1, ..., B$, and $s^0_{j,B+1} = 1 - \sum_{k=1}^{N} s^0_{jk}$ where $\overline{\rho} := \lambda/N\mu$ is the average system utilization. Under the independent server approximation, the value of s^0_{jk} corresponds to the probability that all bases preferred to $j(k)$ in node j's ranking are busy but base $j(k)$ is not.

APPENDIX C

DETAILED DEFINITION OF FUNCTION $R(\cdot)$

Let $R(S) := (R(S)_1, R(S)_2, \ldots, R(S)_N)$ where $R(S)_v$, for $v = 1, \ldots, N$ represents the utilization of ambulance v. Denote by $b = b(v)$ the base at which ambulance v is stationed, and by $\lambda_b = \lambda_b(S)$, the total demand offered to this base, as defined by Equation (A.1); recall from Appendix A that $a_{jk} = \sum_{i=k}^{B+1} s_{ji}$. Then, according to the Erlang loss model, the utilization of each of the n_b ambulances stationed at base b equals the average expected utilization

$$R(S)_v = \frac{1}{n_b} \sum_{k=1}^{n_b} k \, P_{\mathcal{E}}(n_b, \lambda_b/\mu, k),$$

where

$$P_{\mathcal{E}}(n, \rho, k) = \frac{\rho^k/k!}{\sum_{j=0}^{n} \rho^j/j!}$$

is the Erlang probability that k out of the n ambulances are busy. a

BIBLIOGRAPHY

[1] Adelman, D. [2004], 'A price-directed approach to stochastic inventory routing', *Operations Research* **52**(4), 499–514.

[2] Adelman, D. [2007], 'Dynamic bid-prices in revenue management', *Operations Research* **55**(4), 647–661.

[3] Adelman, D. and Mersereau, A. J. [2007], 'Relaxations of weakly coupled stochastic dynamic programs', *Operations Research* **to appear**.

[4] Andersson, T. [2005], Decision support tools for dynamic fleet management, PhD thesis, Department of Science and Technology, Linkoepings Universitet, Norrkoeping, Sweden.

[5] Andersson, T. and Vaerband, P. [2007], 'Decision support tools for ambulance dispatch and relocation', *Journal of the Operational Research Society* **58**, 195–201.

[6] Batta, R., Dolan, J. and Krishnamurthy, N. [1989], 'The maximal expected covering location problem: revisited', *Transportation Science* **23**, 277–287.

[7] Berman, O. [1981*a*], 'Dynamic repositioning of indistinguishable service units on transportation networks', *Transportation Science* **15**(2).

[8] Berman, O. [1981*b*], 'Repositioning of distinguishable urban service units on networks', *Computers and Operations Research* **8**, 105–118.

[9] Berman, O. [1981*c*], 'Repositioning of two distinguishable service vehicles on networks', *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-11**(3).

[10] Berman, O. and Larson, R. [1982], 'The median problem with congestion', *Computers and Operations Research* **9**(22), 119–126.

[11] Bertsekas, D. and Shreve, S. [1978], *Stochastic Optimal Control: The Discrete Time Case.*, Academic Press, New York.

[12] Bertsekas, D. and Tsitsiklis, J. [1996], *Neuro-Dynamic Programming*, Athena Scientific, Belmont, Massachusetts.

[13] Brandeau, M. and Larson, R. C. [1986], Extending and applying the hypercube model to deploy ambulances in boston, *in* A. Swersey and E. Ignall, eds, 'Delivery of Urban Services', North Holland.

[14] Brotcorne, L., Laporte, G. and Semet, F. [2003], 'Ambulance location and relocation models', *European Journal of Operations Research* **147**(3), 451–463.

[15] Budge, S., Ingolfsson, A. and Erkut, E. [2007, Forthcoming], 'Approximating vehicle dispatch probabilities for emergency service systems', *Operations Research* .

[16] Church, R. and ReVelle, C. [1974], 'The maximal covering location problem', *Papers of the Regional Science Association* **32**, 101–108.

[17] Daskin, M. [1983], 'A maximal expected covering location model: formulation, properties, and heuristic solution', *Transportation Science* **17**, 48–70.

[18] de Farias, D. P. and Van Roy, B. [2003], 'The linear programming approach to approximate dynamic programming', *Operations Research* **51**(6), 850–865.

[19] de Farias, D. P. and Van Roy, B. [2004], 'On constraint sampling in the linear programming approach to approximate dynamic programming', *Mathematics of Operations Research* **29**(3).

[20] de Farias, D. P. and Van Roy, B. [2006], 'A cost-shaping linear program for average-cost approximate dynamic programming with performance guarantees', *Mathematics of Operations Research* **31**(3), 597–620.

[21] Erdogan, G., Erkut, E. and Ingolfsson, A. [2007], 'Ambulance deployment for maximum survival', *Naval research logistics* **55**(1), 42–58.

[22] Farias, V. F. and Van Roy, B. [2006], Tetris: A study of randomized constraint sampling, *in* G. Calafiore and F. Dabbene, eds, 'Probabilistic and Randomized Methods for Design Under Uncertainty', Springer-Verlag.

[23] Farias, V. F. and Van Roy, B. [2007], An approximate dynamic programming approach to network revenue management, Technical report, Stanford University, Department of Electrical Engineering.

[24] Fox, B. [1966], 'Discrete optimization via marginal analysis', *Management Science* **13**(3), 210–216.

[25] Gendreau, M., Laporte, G. and Semet, S. [2001], 'A dynamic model and parallel tabu search heuristic for real time ambulance relocation', *Parallel Computing* **27**, 1641–1653.

[26] Gendreau, M., Laporte, G. and Semet, S. [2006], 'The maximal expected coverage relocation problem for emergency vehicles', *Journal of the Operational Research Society* **57**, 22–28.

[27] Goldberg, J. B. [2004], 'Operations research models for the deployment of emergency services vehicles', *Emergency Medical Services Management Journal* **1**(1), 20–39.

[28] Goldberg, J. B. [2007]. Personal Communication.

[29] Goldberg, J. B. and Paz, L. [1991], 'Locating emergency vehicle bases when service time depends on call location', *Transportation Science* **25**(4), 264–280.

[30] Goldberg, J. B. and Szidarovsky, F. [1991], 'Methods for solving nonlinear equations used in evaluating emergency vehicle busy probabilities', *Operations Research* **39**(6), 903–916.

[31] Green, L. V. and Kolesar, P. J. [2004], 'Improving emergency responsiveness with management science', *Management Science* **50**(8), 1001–1014.

[32] Gross, D. and Harris, C. M. [1985], *Fundamentals of Queuing Theory*, second edn, Wiley, New York.

[33] Harel, A. [1988], 'Convexity properties of the Erlang loss formula.', *Operations Research* **38**(3), 499–505.

[34] Henderson, S. G. and Mason, A. J. [2004], Ambulance service planning: Simulation and data visualisation, *in* M. L. Brandeau, F. Sainfort and W. P. Pierskalla, eds, 'Operations Research and Health Care: A Handbook of Methods and Applications', Handbooks in Operations Research and Management Science, Kluwer Academic, pp. 77–102.

[35] Ingolfsson, A. [2006], 'The impact of ambulance system status management'. Presentation at 2006 INFORMS Conference.

[36] Ingolfsson, A., Erkut, E. and Budge, S. [2003], 'Simulation of single start station for Edmonton EMS', *Journal of the Operational Research Society* **54**(7), 736–746.

[37] Jagers, A. A. and van Doorn, E. A. [1986], 'On the continued Erlang loss function', *Operations Research Letters* **5**(1), 43–46.

[38] Jarvis, J. [1975], Optimization in stochastic systems with distinguishable servers, Technical report, Operations Research Center, M.I.T.

[39] Jarvis, J. [1985], 'Approximating the equilibrium behavior of multi-server loss systems', *Management Science* **31**, 235–239.

[40] Klabjan, D. and Adelman, D. [2007], 'A convergent infinite dimensional linear programming algorithm for deterministic semi-markov decision processes on borel spaces', *Mathematics of Operations Research* **32**(3), 528–550.

[41] Kolesar, P. and Walker, W. E. [1974], 'An algorithm for the dynamic relocation of fire companies', *Operations Research* **22**(2), 249–274.

[42] Koole, G. and Talim, J. [2000], Exponential approximation of multi-skill call center architectures, *in* 'Proceedings of QNETs 2000', QNETs, Ilkley (UK). pages 23/1-10.

[43] Larson, R. C. [1974], 'A hypercube queuing model for facility location and redistricting in urban emergency services', *Computers and Operations Research* **1**(1), 67–95.

[44] Larson, R. C. [1975], 'Approximating the performance of urban emergency systems', *Operations Research* **23**(5), 845–868.

[45] Manne, A. [1960], 'Linear prgramming and sequential decisions', *Management Science* **6**(3), 259–267.

[46] Marianov, V. and ReVelle, C. [1994], 'The queuing probabilistic location set covering problem and some extensions', *Socio-Economic Planning Sciences* pp. 167–178.

[47] Nair, R. and Miller-Hooks, E. [2006], 'A case study of ambulance location and relocation'. Presentation at 2006 INFORMS Conference.

[48] Powell, W. B. [2007], *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, John Wiley & Sons, Hoboken, NJ.

[49] Restrepo, M., Henderson, S. and Topaloglu, H. [2007], Erlang loss models for the static deployment of ambulances. Submitted for publication.

[50] ReVelle, C. and Hogan, K. [1989], 'The maximum reliability location problem and α-reliable p-center problem: derivatives of the probabilistic location set covering problems', *Annals of Operations Research* **18**, 155–174.

[51] Richards, D. P. [2007], Optimised ambulance redeployment strategies, Master's thesis, Department of Engineering Science, University of Auckland, Auckland, New Zealand.

[52] Schweitzer, P. and Seidmann, A. [1985], 'Generalized polynomial approximations in Markovian decision processes', *Journal of Mathematical Analysis and Applications* **110**, 568–582.

[53] Si, J., Barto, A. G., Powell, W. B. and Wunsch II, D., eds [2004], *Handbook of Learning and Approximate Dynamic Programming*, Wiley-Interscience, Piscataway, NJ.

[54] Singer, M. and Donoso, P. [2008], 'Assesing an ambulance service with queueing theory', *Computers and Operations Research* **35**, 2549–2560.

[55] Sutton, R. S. [1988], 'Learning to predict by the methods of temporal differences', *Machine Learning* **3**, 9–44.

[56] Swersey, A. J. [1994], The deployment of police, fire and emergency medical units, *in* S. M. Pollock, M. Rothkopf and A. Barnett, eds, 'Operations Research and the Public Sector', Vol. 6 of *Handbooks in Operations Research and Management Science*, North-Holland, pp. 151–190.

[57] Topaloglu, H. and Powell, W. B. [2006], 'Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems', *INFORMS Journal on Computing* **18**(1), 31–42.

[58] Toregas, G., Saqin, R., ReVelle, C. and Berman, L. [1971], 'The location of emergency service facilities', *Operations Research* **19**(6), 1363–1373.

[59] Tsitsiklis, J. N. [1994], 'Asynchronous stochastic approximation and *Q*-learning', *Machine Learning* **16**, 185–202.

[60] Tsitsiklis, J. and Van Roy, B. [1997], 'An analysis of temporal-difference learning with function approximation', *IEEE Transactions on Automatic Control* **42**, 674–690.

[61] Tsitsiklis, J. and Van Roy, B. [2001], 'Regression methods for pricing complex American-style options', *IEEE Transactions on Neural Networks* **12**(4), 694–703.

[62] Van Roy, B., Bertsekas, D. P., Lee, Y. and Tsitsiklis, J. N. [1997], A neuro dynamic programming approach to retailer inventory management, *in* 'Proceedings of the IEEE Conference on Decision and Control'.

[63] Watkins, C. J. C. H. and Dayan, P. [1992], '*Q*-learning', *Machine Learning* **8**, 279–292.

[64] Wolff, R. W. [1989], *Stochastic modeling and the theory of queues*, Prentice-Hall, Englewood Cliffs, NJ.

[65] Yan, X., Diaconis, P., Rusmevichientong, P. and Van Roy, B. [2005], 'Solitaire: Man versus machine', *Advances in Neural Information Processing Systems* **17**.